МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

__ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: ***«Розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe»***

Виконав: студент групи 4151

_____Островський Ю. С.

(підпис, ПІБ)

Керівник роботи:

Доцент. к.т.н., доцент_____

(посада, науковий ступень вчене звання)

_____ ___Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Островський Юрій Сергійович
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe
Керівник роботи Пухалевич А. В.
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р.
3. Вихідні дані по роботі: _____
4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).
5. Перелік презентаційних матеріалів 1. Титульний слайд; 2. Мета та завдання кваліфікаційної роботи; 3. Аналіз вимог до програмного забезпечення; 4. Архітектура програмного забезпечення; 5. Діаграма варіантів використання; 6. Діаграма діяльності; 7. Діаграма класів; 8. Діаграма послідовності; 9. Логічна модель даних; 10. Фізична модель даних; 11. Результати розробки; 12. Висновки.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

6. Консультанти розділів роботи

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Островський Ю. С.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційну роботу присвячено розв’язанню практичної задачі інженерії програмного забезпечення, а саме — розробці програмного забезпечення для автоматизації нарахування заробітної плати в кав’ярні Lite Cafe.

У ході виконання роботи було проведено аналіз практичної задачі, яка виникає в умовах змінного графіка працівників та потребує точної фіксації робочого часу і розрахунку заробітної плати. Також було досліджено приклади аналогічного програмного забезпечення, виявлено їхні недоліки та обґрунтовано доцільність створення власного рішення.

На основі цього сформульовано постановку задачі з урахуванням потреб малого бізнесу та особливостей функціонування кав’ярні. Було спроєктовано архітектуру програмного забезпечення на основі клієнт-серверної моделі з використанням принципів об’єктно-орієнтованого програмування та шаблону MVC.

Реалізовано функціональні модулі програмного забезпечення: облік відвідувань працівників за допомогою QR-кодів, розрахунок заробітної плати з урахуванням податків, премій і штрафів, а також мобільної частини для фіксації робочого часу. Проведено тестування програмного забезпечення, яке підтвердило його працездатність та відповідність вимогам.

Робота виконана на 101 сторінці, містить 33 рисунки, 17 таблиць, 5 додатків та 14 використаних джерел.

ABSTRACT

The qualification work is dedicated to solving a practical problem of software engineering, namely - the development of software for automating payroll in the coffee shop Lite Cafe.

During the work, an analysis of a practical problem was carried out, which arises in conditions of a variable schedule of employees and requires accurate recording of working hours and calculation of wages. Examples of similar software were also studied, their shortcomings were identified and the feasibility of creating your own solution was substantiated.

Based on this, the problem statement was formulated, taking into account the needs of small businesses and the peculiarities of the functioning of the coffee shop. The software architecture was designed based on the client-server model using the principles of object-oriented programming and the MVC template.

Functional software modules implemented: accounting for employee attendance using QR codes, calculation of wages taking into account taxes, bonuses and fines, as well as a mobile part for recording working hours. The software was tested, which confirmed its functionality and compliance with the requirements.

The work is 101 pages long, contains 33 figures, 17 tables, 5 appendices and 14 sources used.

ЗМІСТ

ВСТУП

У цій кваліфікаційній роботі розглядається практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe.

Необхідність створення подібного програмного забезпечення обумовлена практичними потребами Lite Cafe у спрощенні бухгалтерських процесів. Аналогічним програмним забезпеченням є популярна програма Shifton, яка частково реалізує схожі функції, проте не повністю адаптована під умови невеликих закладів, таких як Lite Cafe. Тому є доцільною розробка власного розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe. Це дозволить автоматизувати розрахунок заробітної плати з урахуванням ведення робочого часу, премій, податків та інших відрахувань.

Метою роботи є розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe.

Завдання, які необхідно вирішити у кваліфікаційній роботі:

1) Провести аналіз практичної задачі інженерії програмного забезпечення автоматизації нарахування заробітної плати кав’ярні Lite Cafe;
2) Провести аналіз аналогічного існуючого програмного забезпечення для автоматизації нарахування заробітної плати;
3) Виконати постановку задачі на розробку програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe;
4) Спроєктувати архітектуру програмного забезпечення відповідно до визначених вимог;

5) Реалізувати функціональні модулі програмного забезпечення: розрахунок зарплати, ведення робочого часу та телефонну частину для працівників за допомогою якого, будуть вести фіксація робочого часу;
6) Провести тестування програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe.

Об’єктом дослідження у кваліфікаційній роботі є процес розробки програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

1.1 Аналіз практичної задачі інженерії програмного забезпечення автоматизації нарахування заробітної плати кав'ярні Lite Cafe

Практична задача інженерії програмного забезпечення, що розглядається у цій роботі, характеризується комплексністю та невизначеністю умов, а саме — розробка програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe. Ця задача виникає через необхідність точно вести облік робочого часу працівників, розраховувати зарплату, податкових нарахувань, премій та відрахувань.

У кав'ярні Lite Cafe працюють співробітники за змінним графіком. Часто трапляються випадки, коли хтось бере відпустку, змінює зміну або підміняє іншого працівника. Через це складно вручну вести точний облік робочого часу і здійснювати розрахунок заробітної плати. Власник кав'ярні витрачає значну кількість часу на підрахунки, що підвищує ризик помилок і знижує ефективність управління.

Через це існує необхідність в автоматизації процесу обліку відпрацьованих годин та розрахунку заробітної плати — індивідуальних ставок, податків, штрафів, премій тощо. Це дозволить зменшити навантаження на власника та забезпечить точність і оперативність фінансових розрахунків.

Подібні задачі вирішуються шляхом створення спеціалізованих програмних рішень, що розробляються з використанням сучасних мов

програмування (наприклад: Java, JavaScript, html), систем керування базами даних (наприклад, PostgreSQL, MariaDB), та фреймворків для вебінтерфейсів і мобільних застосунків (наприклад: Android Studio). Саме використання таких інструментів дозволяє досягти гнучкості та зручності в програмі.

Вибір зазначених технологій узгоджується з рекомендаціями щодо інженерії програмного забезпечення, наведеними у працях [1, 2].

1.2 Аналіз аналогічного існуючого програмного забезпечення для автоматизації нарахування заробітної плати

У процесі дослідження було розглянуто приклад існуючої програми Shifton (рисунок 1.1), яка пропонує рішення для планування змін і обліку робочого часу. Програма дозволяє формувати графіки, обліковувати години, створювати звіти [3]. Проте її функціонал переважно орієнтований на великі організації, і не охоплює повний спектр задач, які стоять перед невеликими закладами, як-от Lite Cafe. Зокрема, Shifton не реалізує автоматичний розрахунок заробітної плати з урахуванням умов закладу, податкових відрахувань.

Також, аналогічним програмним забезпеченням є Paychex (рисунок 1.2). Програма є комплексним рішенням для нарахування заробітної плати, ведення кадрового обліку та управління персоналом [4]. Проте, програмне забезпечення Paychex, не дозволяє гнучко адаптувати алгоритми розрахунку зарплати під локальні особливості, зокрема нарахування бонусів, штрафів, які часто зустрічаються у малому бізнесі.

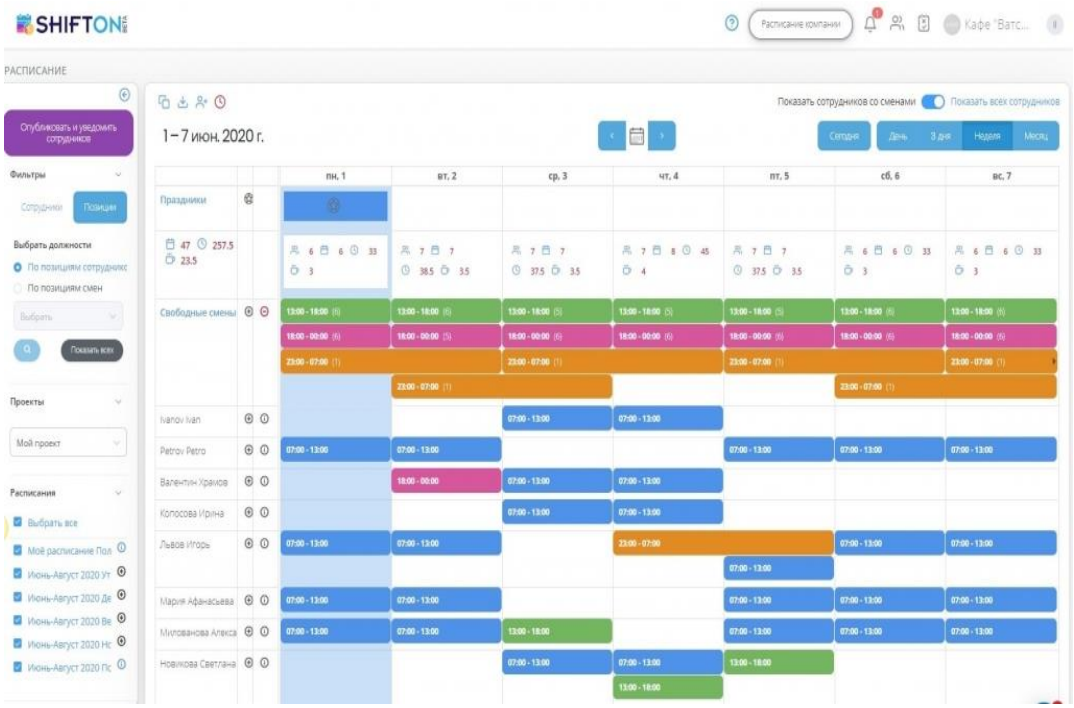

Рисунок 1.1 – Інтерфейс програми Shifton

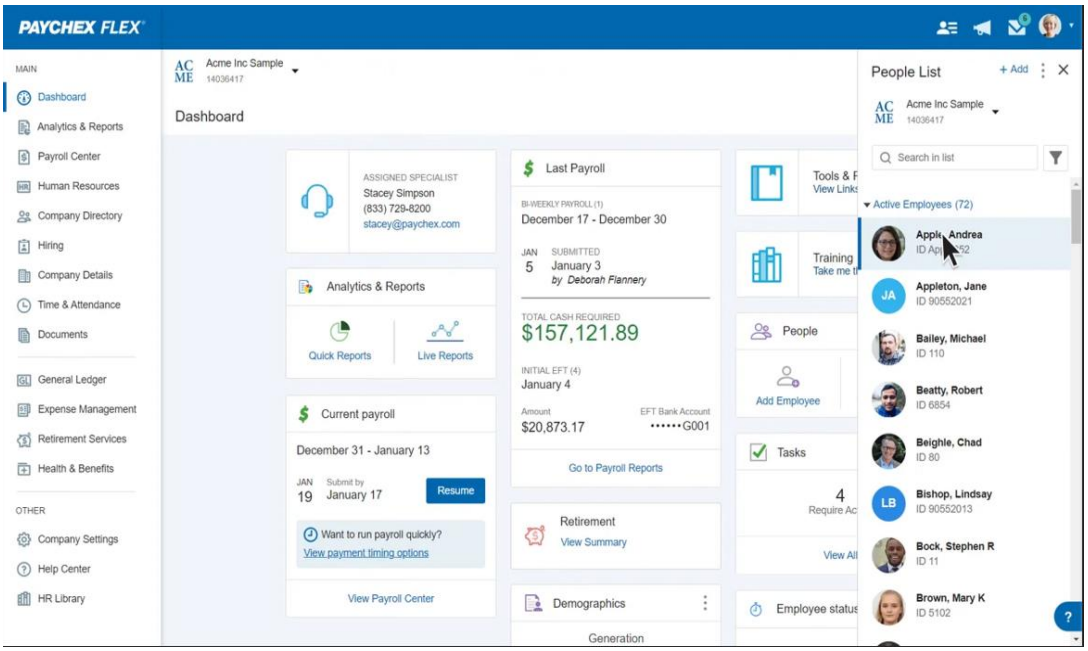

Рисунок 1.2 - Інтерфейс програми Paychex

Це обґрунтовує необхідність створення власного програмного продукту, адаптованого до конкретних потреб замовника, з урахуванням особливостей роботи кав’ярні.

1.3 Постановка задачі на розробку програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe

Відповідно до проведеного аналізу практичної задачі інженерії програмного забезпечення, постановка задачі кваліфікаційної роботи є наступною: необхідно розробити програмне забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe.

Вимоги до складу виконуваних функцій

1) Введення та редагування даних працівника – можливість додавання, зміни та видалення інформації про працівників: ПІБ, номер телефону, електрона адреса, фото, місце проживання та попередній досвід роботи;
2) Фіксація приходу на роботу працівника;
3) Фіксація виходу з роботи працівника;
4) Нарахування заробітної плати з урахуванням податків, премій та штрафів. Нарахування зарплати за погодинною або денною ставкою з урахуванням податкових відрахувань, бонусів і штрафів.
5) Розблокування програмного забезпечення для адміністратора
6) Блокування програмного забезпечення для адміністратора – PIN-код доступу, можливість блокування програми на випадок відходу адміністратора від робочого місця, обмеження стороннього втручання до системи.

Вимоги до організації вхідних та вихідних даних:

Для введення та редагування даних працівника: вхідними даними є ПІБ, номер телефону, фото, електрона адреса, місце проживання, попередній досвід роботи. Вихідними даними є збережені та оновлені записи працівників у базі.

Для фіксації приходу на роботу працівника: вхідними даними є унікальний ключ працівника. Вихідними даними є час приходу.

Для фіксації виходу з роботи працівника: вхідними даними є унікальний ключ працівника. Вихідними даними є час виходу.

Для розрахунків заробітної плати з урахуванням податків, премій та штрафів: вхідними даними є дані входів та виходів з роботи, ставка, коефіцієнти податків, розміри премій і штрафів. Вихідними даними є розрахована сума до виплати.

Для ідентифікації телефонної частини програмного забезпечення, призначений для працівників: вхідними даними є ідентифікаційний ключ або QR-код для ідентифікації. Вихідними даними є перегляд особистих даних працівників.

Для розблокування програмного забезпечення для адміністратора: вхідними даними є PIN-код, електрона адреса адміністратора, резервний PIN-код. Вихідними даними є розблокований доступ.

Для блокування програмного забезпечення для адміністратора: вхідними даними є PIN-код. Вихідними даними є заблокований доступ.

Серед нефункціональних вимог варто зазначити: зручність інтерфейсу, стабільність роботи, сумісність із популярними операційними системами, а також збереження конфіденційності даних персоналу.

Розробка програмного забезпечення для автоматизації нарахування заробітної плати дозволить зменшити ручну працю адміністратора, мінімізувати помилки та підвищити ефективність обліку в кав’ярні Lite Cafe.

Більш докладний опис постановки завдання подано в технічному завданні, що описано в додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

2.1 Аналіз вимог до програмного забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe

2.1.1 Діаграма варіантів використання

На основі постановки задачі прикладів аналогічного програмного забезпечення та потреб кав'ярні Lite Cafe, було сформульовано перелік вимог до майбутньої системи представлено в діаграмі варіантів використання (рисунок 2.1).

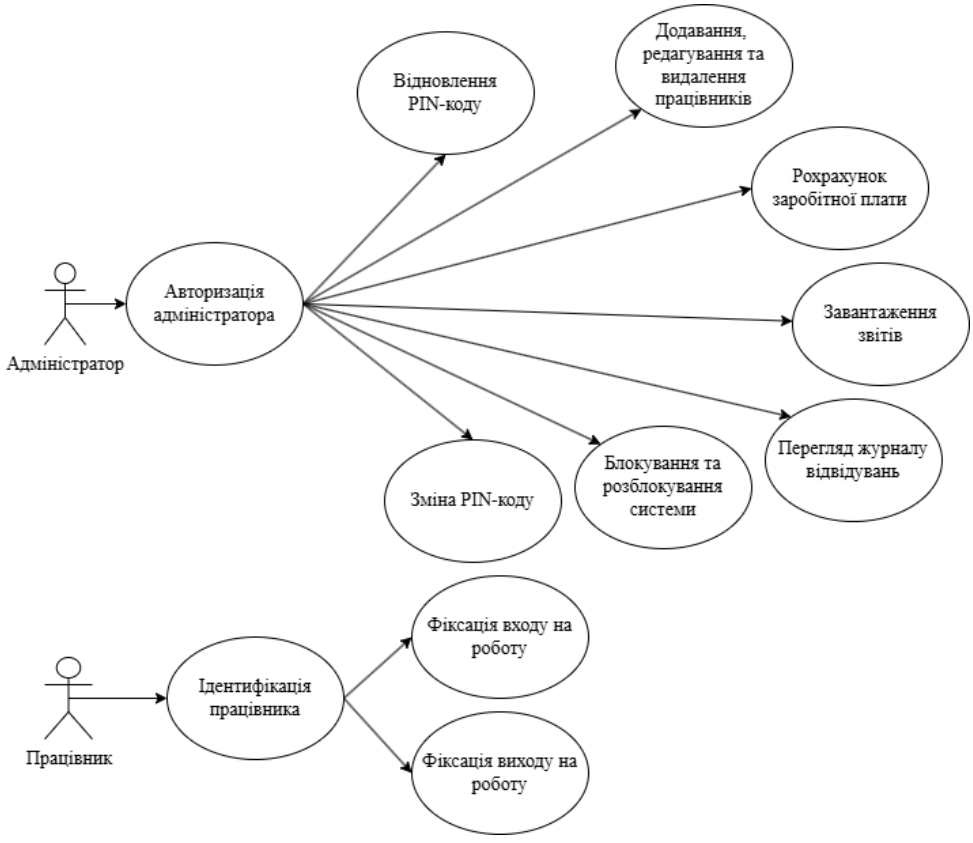

Рисунок 2.1 – Діаграма варіантів використання

При формуванні вимог враховувались рекомендації стандарту ISO/IEC 25010 щодо якості програмного забезпечення [5]. Проведений аналіз діаграми варіантів використання [6] дозволяє перейти до детального розгляду кожного з них. У таблиці 2.1 подано узагальнений перелік усіх варіантів використання.

Таблиця 2.1 – Варіанти використання та їх короткий опис

№	Назва варіанта використання	Актор	Формулювання
1	Авторизація адміністратора	Адміністратор	Надання адміністратору доступу до функцій системи програмного забезпечення після авторизації.
2	Відновлення PIN-коду	Адміністратор	Можливість відновлення PIN-коду через електронну пошту.
3	Зміна PIN-коду	Адміністратор	Заміна чинного PIN-коду на новий після авторизації.
4	Блокування та розблокування системи	Адміністратор	Тимчасове обмеження доступу до системи та подальше його відновлення.
5	Додавання, редагування та видалення працівників	Адміністратор	Управління інформацією про працівників системи.
6	Розрахунок заробітної плати	Адміністратор	Автоматичне обчислення заробітної плати працівників.
7	Завантаження звітів	Адміністратор	Формування та експорт фінансових звітів.
8	Перегляд журналу відвідувань	Адміністратор	Перегляд історії відвідувань працівників.
9	Ідентифікація працівника	Працівник	Розпізнавання працівника за допомогою QR-коду або іншого ідентифікатора.
10	Фіксація входу на роботу	Працівник	Реєстрація початку робочої зміни.
11	Фіксація виходу з роботи	Працівник	Реєстрація завершення робочої зміни.

2.1.2 Специфікації варіантів використання

Для конкретизації функціональних вимог до системи були розроблені детальні специфікації варіантів використання. Детальний опис кожного з варіантів наведено у таблицях 2.2–2.12.

Таблиця 2.2 – Прецедент використання «Авторизація адміністратора»

Складова	Опис
1	2
Назва прецеденту	Авторизація адміністратора
Короткий опис	Забезпечує вхід адміністратора до системи через введення PIN-коду або підтвердження електронної пошти
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Адміністратор відкриває вікно авторизації 2. Вводить PIN-код 3. Натискає кнопку “Увійти” 4. Програмне забезпечення перевіряє правильність PIN-коду 5. У разі успіху відкривається головна сторінка адміністратора
Передумови	Програмне забезпечення активна, адміністратор зареєстрований і має PIN-код
Пост умови	Адміністратор отримує доступ до функціоналу керування системою

Кінець таблиці 2.2

Складова	Опис
1	2
Альтернативний потік	1. Введено неправильний PIN-код – програмне забезпечення виводить повідомлення про помилку 2. PIN втрачено – адміністратор обирає “Відновити PIN” та слідує інструкціям для відновлення через електронну адресу

Таблиця 2.3 – Прецедент використання «Відновлення PIN-коду»

Складова	Опис
1	2
Назва прецеденту	Відновлення PIN-коду
Короткий опис	Дозволяє адміністратору відновити PIN-код за допомогою електронна адреса
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Обирає “Забули PIN?” 2. Вводить електронну адресу 3. Отримує лист з посиланням 4. Задає новий PIN
Передумови	Підтвердження електронної пошти у програмному забезпеченні
Пост умови	Створено новий PIN-код
Альтернативний потік	Електронна адреса не підтверджена — програмне забезпечення блокує відновлення

Таблиця 2.4 – Прецедент використання «Зміна PIN-коду»

Складова	Опис
1	2
Назва прецеденту	Зміна PIN-коду
Короткий опис	Дозволяє адміністратору змінити свій PIN-код
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Вводить старий PIN 2. Вводить новий PIN 3. Підтверджує
Передумови	Адміністратор авторизований
Пост умови	PIN-код оновлено у системі
Альтернативний потік	Старий PIN невірний – програмне забезпечення повідомляє про помилку

Таблиця 2.5 – Прецедент використання «Блокування та розблокування системи»

Складова	Опис
1	2
Назва прецеденту	Блокування та розблокування системи
Короткий опис	Тимчасове обмеження доступу до системи
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Натискає кнопку “Блокувати” 2. Для відновлення — вводить PIN
Передумови	Програмне забезпечення в активному режимі
Пост умови	Доступ закрито або відкрито

Кінець таблиці 2.5

Складова	Опис
1	2
Альтернативний потік	PIN при розблокуванні невірний — доступ не надано

Таблиця 2.6 – Прецедент використання «Перегляд журналу відвідувань»

Складова	Опис
1	2
Назва прецеденту	Перегляд журналу відвідувань
Короткий опис	Надання адміністратору можливості переглядати історію відвідувань працівників
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Адміністратор натискає “Журнал відвідувань” 2. Програмне забезпечення відображає список записів (дати, часи, імена)
Передумови	Адміністратор авторизований, програмне забезпечення має дані про відвідування
Пост умови	Журнал успішно відображено
Альтернативний потік	Відсутні записи — програмне забезпечення повідомляє, що журнал порожній

Таблиця 2.7 – Прецедент використання «Управління працівниками»

Складова	Опис
1	2
Назва прецеденту	Управління працівниками

Кінець таблиці 2.7

Короткий опис	Забезпечує додавання, редагування або видалення працівників
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Обирає функцію “Працівники” 2. Додає або редагує дані 3. Зберігає зміни
Передумови	Адміністратор авторизований
Пост умови	Дані працівника додано, змінено або видалено
Альтернативний потік	Введено неповні/некоректні дані — програмне забезпечення повідомляє про помилку

Таблиця 2.8 – Прецедент використання «Розрахунок заробітної плати»

Складова	Опис
1	2
Назва прецеденту	Розрахунок заробітної плати
Короткий опис	Автоматизує обчислення заробітної плати
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Обирає працівника 2. Вказує період 3. Програмне забезпечення розраховує зарплату
Передумови	Дані про присутність та ставки наявні
Пост умови	Отримано результат розрахунку

Кінець таблиці 2.8

Складова	Опис
1	2
Альтернативний потік	Відсутні дані про відпрацьовані години — програмне забезпечення повідомляє про неможливість розрахунку

Таблиця 2.9 – Прецедент використання «Завантаження звітів»

Складова	Опис
1	2
Назва прецеденту	Завантаження звітів
Короткий опис	Експорт фінансової інформації у форматі PDF або таблиці
Головний актор	Адміністратор
Другорядні актори	Програмне забезпечення
Базовий потік	1. Обирає звіт 2. Натискає “Завантажити” 3. Зберігає файл
Передумови	Є результати розрахунків зарплати
Пост умови	Звіт завантажено у локальну систему
Альтернативний потік	Немає сформованих звітів — програмне забезпечення показує повідомлення

Таблиця 2.10 – Прецедент використання «Ідентифікація працівника»

Складова	Опис
1	2
Назва прецеденту	Ідентифікація працівника
Короткий опис	Розпізнавання працівника для доступу до персоналізованих дій

Кінець таблиці 2.10

Складова	Опис
1	2
Головний актор	Працівник
Другорядні актори	Програмне забезпечення
Базовий потік	1. Працівник відкриває мобільний застосунок 2. Сканує QR-код або вводить код 3. Програмне забезпечення перевіряє дані
Передумови	Працівник зареєстрований у системі
Пост умови	Програмне забезпечення розпізнає працівника
Альтернативний потік	Невірний код — програмне забезпечення повідомляє про помилку

Таблиця 2.11 – Прецедент використання «Фіксація входу на роботу»

Складова	Опис
1	2
Назва прецеденту	Фіксація входу на роботу
Короткий опис	Реєстрація початку робочого дня працівником
Головний актор	Працівник
Другорядні актори	Програмне забезпечення
Базовий потік	1. Після ідентифікації працівник натискає “Прихід” і сканує QR-код 2. Програмне забезпечення фіксує час
Передумови	Працівник ідентифікований
Пост умови	Запис про початок зміни збережено у базі

Кінець таблиці 2.11

Складова	Опис
1	2
Альтернативний потік	Сканування не розпізнано — програмне забезпечення просить повторити

Таблиця 2.12 – Прецедент використання «Фіксація виходу з роботи»

Складова	Опис
1	2
Назва прецеденту	Фіксація виходу з роботи
Короткий опис	Реєстрація завершення робочої зміни
Головний актор	Працівник
Другорядні актори	Програмне забезпечення
Базовий потік	1. Працівник натискає “Вихід” і сканує QR-код 2. Програмне забезпечення фіксує час
Передумови	Працівник ідентифікований, був зафіксований на вході
Пост умови	Запис про завершення зміни збережено
Альтернативний потік	Попередній вхід не зафіксовано — програмне забезпечення показує попередження

2.2 Архітектура програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe

Після аналізу вимог було розроблено архітектуру програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe. Програмне забезпечення побудована за клієнт-серверною архітектурою, де серверна частина відповідає за обробку логіки, збереження даних та взаємодію з базою даних, а клієнтська частина представлена у вигляді вебінтерфейсу для адміністратора та мобільної частини для працівників.

Основні компоненти програмного забезпечення взаємодіють через вебінтерфейс та мобільну частину, які відповідають за ідентифікацію працівників, фіксацію часу входу та виходу з роботи, а також управління даними працівників та нарахування заробітної плати з вебінтерфейсу. Мобільна частина програмного забезпечення підтримує сканування QR-кодів для фіксації часу, що значно спрощує процес контролю відвідуваності.

У реалізації використано об’єктно-орієнтоване програмування (ООП), що дозволить чітко структурувати програмні компоненти та розділити їхні обов’язки [7, 8].

Архітектура побудована за принципом MVC (Model-View-Controller). Це дало змогу розділити логіку обробки даних (Model), управління запитами (Controller) та відображення інформації (View), що зробило систему більш гнучкою для розширення та зручною для підтримки.

Усі компоненти взаємодіють між собою через REST API, що забезпечить швидку обробку запитів, гнучкість використання та можливість розширення системи в майбутньому.

При моделюванні функціоналу враховувався досвід впровадження подібних рішень у сервісах Shifton та Paychex, однак розроблене рішення

адаптоване під потреби малого бізнесу та конкретно під формат роботи кав’ярні Lite Cafe.

2.3 Проєкт програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe.

2.3.1 Ескізний проєкт програмного забезпечення для автоматизації нарахування заробітної плати кав’ярні Lite Cafe.

Ескіз інтерфейсу адміністративної частини та мобільної частини програмного забезпечення подано на рисунках 2.2-2.4.

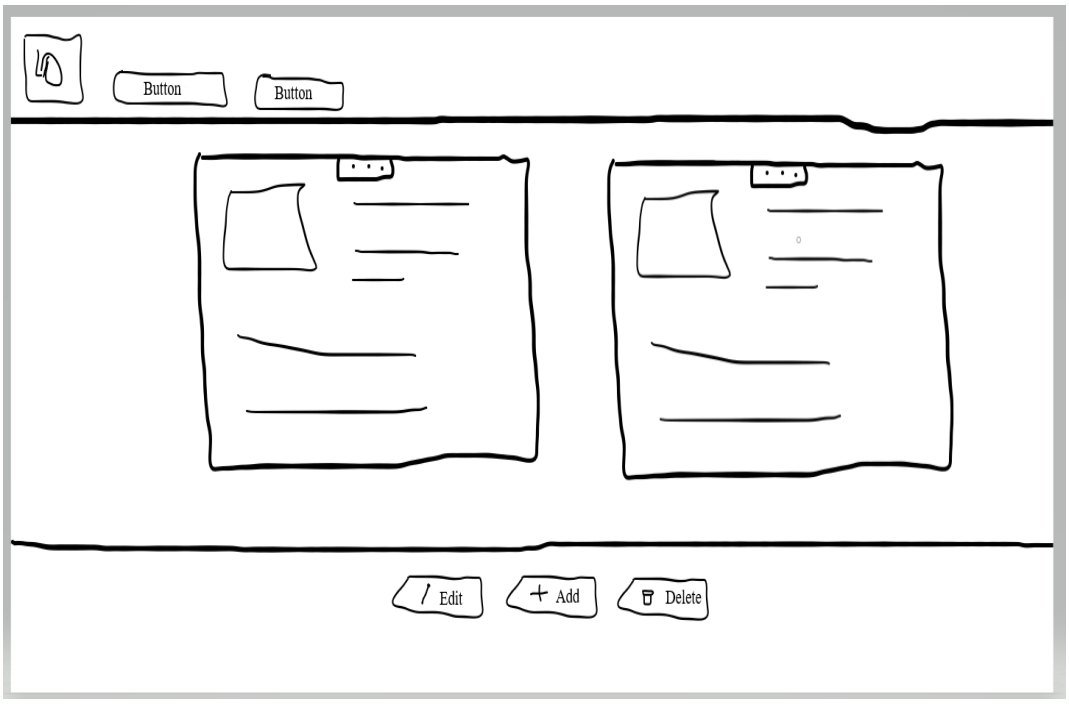

Рисунок 2.2 – Ескіз інтерфейсу відділу кадрів

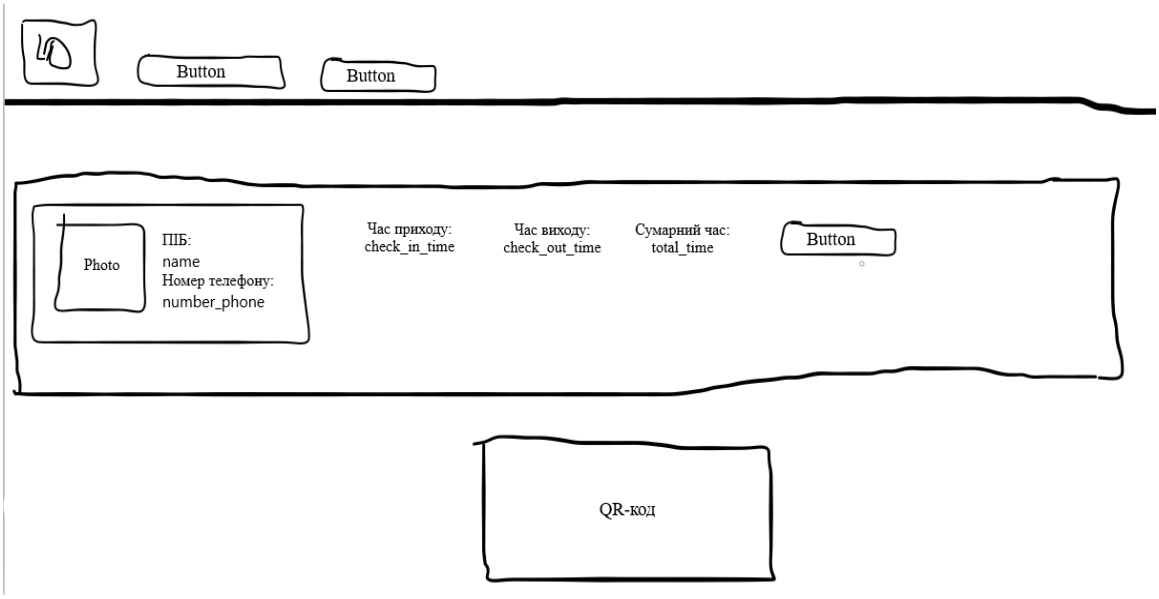

Рисунок 2.3 – Ескіз інтерфейсу фінансового відділу

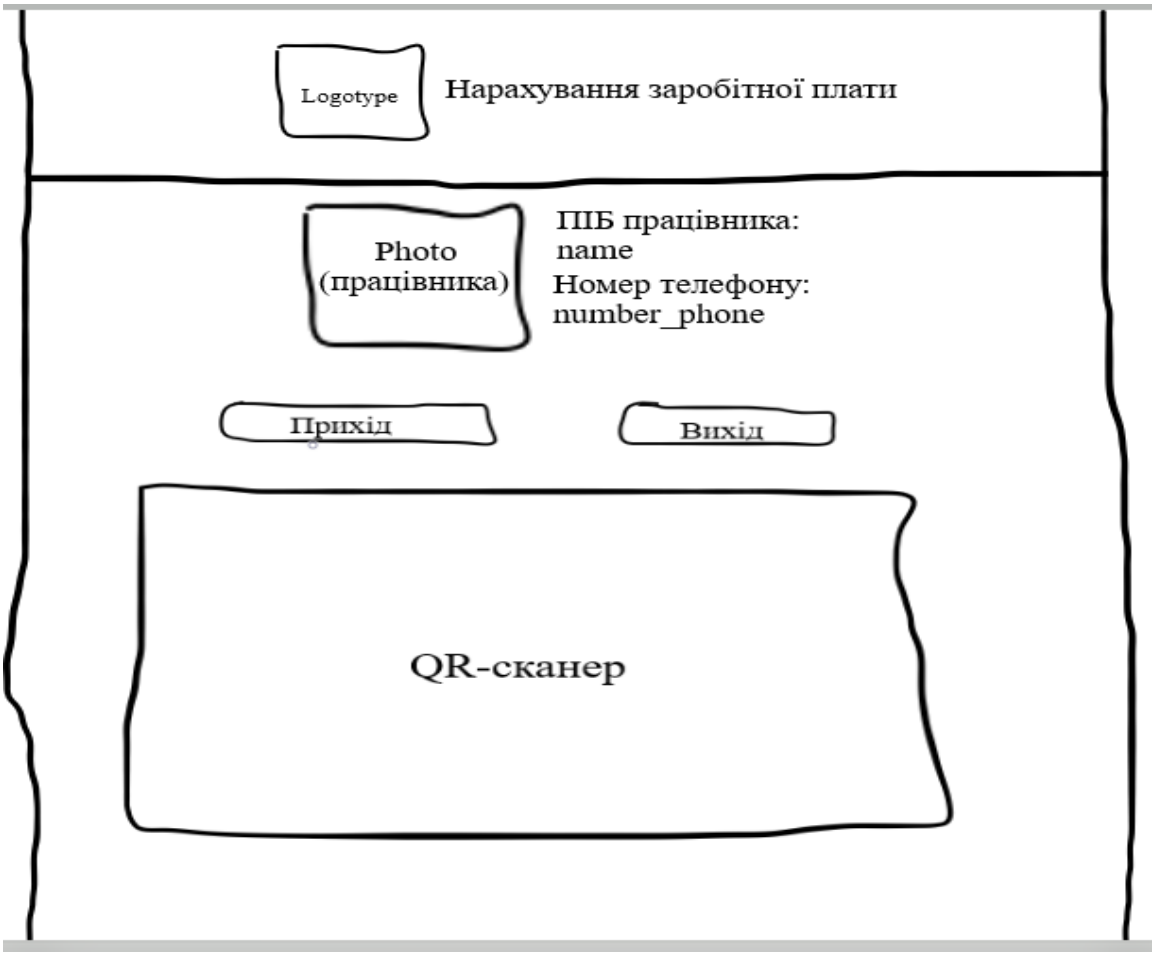

Рисунок 2.4 – Ескіз мобільної частини

2.3.2 Технічний проєкт програмного забезпечення автоматизації нарахування заробітної плати

Під час розробки були створені діаграма класів та логічна модель даних. Після аналізу класів програмного забезпечення була підготовлена специфікація класів.

Діаграма класів

Діаграми класів [9] програмного забезпечення представлена на рисунку 2.5.

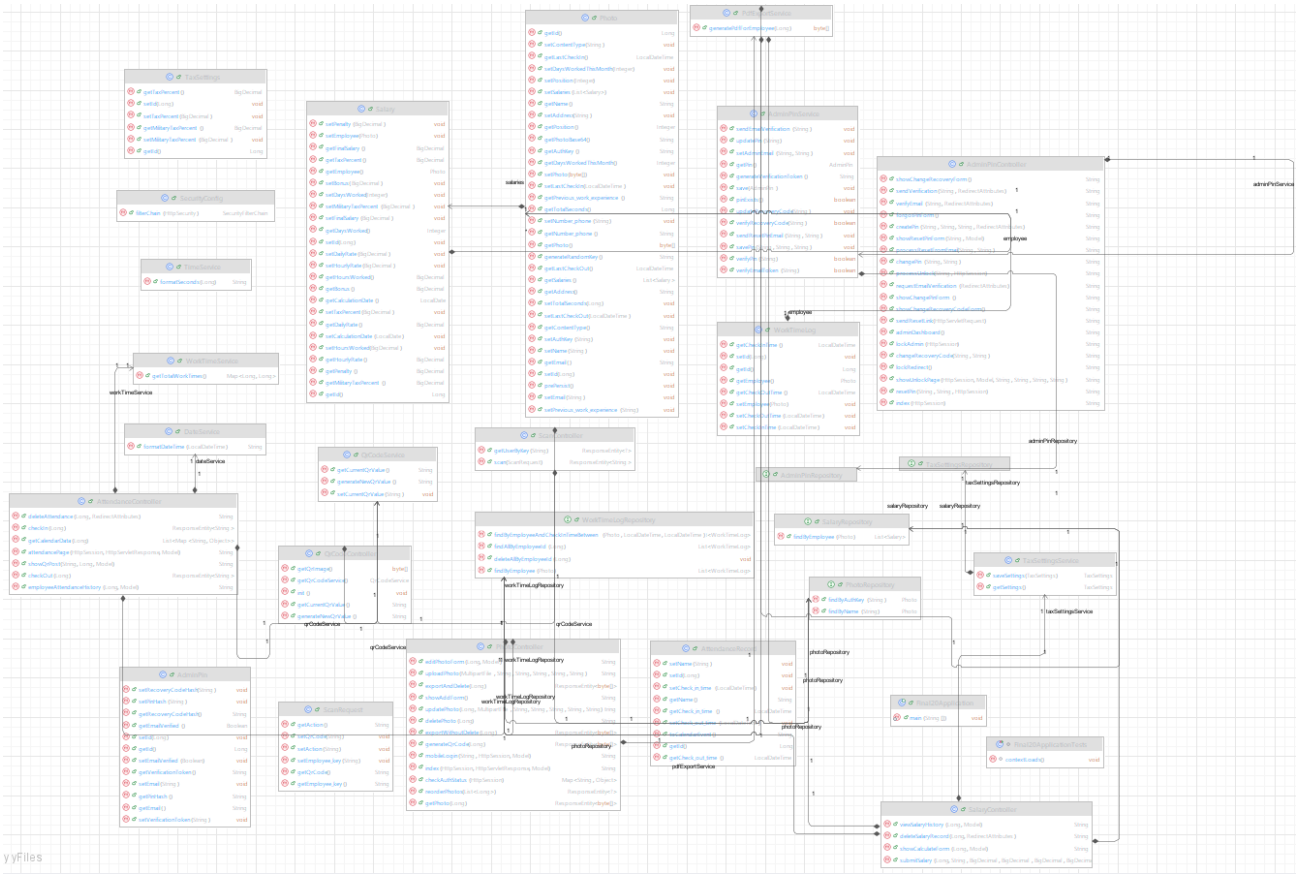

Рисунок 2.5 – Діаграма класів програмного забезпечення

Специфікація класів

Після аналізу класів шляхом побудови діаграми класів, можна сформувати специфікацію основних класів. Нижче наведено специфікацію класів.

Назва: AdminPinService

Відповідальність класу: Керує логікою роботи з PIN-кодами адміністраторів.

Функції, які надає клас:

- saveAdminPin() – зберігає або оновлює PIN для адміністратора;
- checkAdminPin() – перевіряє, чи PIN відповідає збереженому;
- getAdminPin() – отримує поточний PIN.

Назва: SalaryService

Відповідальність класу: Розраховує та надає дані про заробітну плату.

Функції, які надає клас:

- calculateSalary() – розраховує зарплату з урахуванням робочого часу та податків;
- getAllSalaries() – повертає список всіх зарплат;
- getSalaryById() – отримує зарплату за ідентифікатором.

Назва: TaxSettingService

Відповідальність класу: Надає логіку для управління податковими налаштуваннями.

Функції, які надає клас:

- getTaxSettings() – отримання всіх податкових правил;
- updateTaxSettings() – оновлення податкових параметрів.

Назва: PhotoService

Відповідальність класу: Керує логікою збереження, отримання та обробки фото.

Функції, які надає клас:

- uploadPhoto() – завантажує фото користувача;
- getPhotoById() – отримує фото за ID;
- deletePhoto() – видаляє фото.

Назва: AttendanceController

Відповідальність класу: Обробляє HTTP-запити для обліку присутності.

Функції, які надає клас:

- markAttendance() – відмічає присутність на основі QR-коду;
- getAttendanceRecords() – повертає історію відвідувань.

Назва: ScanController

Відповідальність класу: Приймає запити зі сканерів QR-кодів.

Функції, які надає клас:

- scanQrCode() – розшифровує QR-код;
- logScanEvent() – фіксує сканування у базі.

Назва: QRCodeService

Відповідальність класу: Генерує та обслуговує QR-коди.

Функції, які надає клас:

- generateQrCode() – створює QR-код для користувача;
- validateQrCode() – перевіряє достовірність сканованого коду.

Назва: WorkTimeService

Відповідальність класу: Керує логікою підрахунку робочого часу.

Функції, які надає клас:

- saveWorkTime() – зберігає зміну користувача;
- calculateTotalTime() – рахує сумарний час роботи за період.

Назва: SecurityConfig

Відповідальність класу: Визначає налаштування безпеки застосунку.

Функції, які надає клас:

- configureHttpSecurity() – задає правила доступу;

- configureAuthentication() – підключає механізми автентифікації.

Назва: SalaryController

Відповідальність класу: Обробляє запити, пов’язані із заробітною платою.

Функції, які надає клас:

- getAllSalaries() – повертає всі записи про зарплати;
- getSalaryById() – отримує зарплату за ID;
- generateSalaryReport() – створює звіт по зарплаті.

Назва: TaxSettingsController

Відповідальність класу: Керує запитами щодо податкових налаштувань.

Функції, які надає клас:

- getTaxSettings() – отримує поточні податкові налаштування;
- updateTaxSettings() – оновлює відсотки податків.

Назва: PhotoController

Відповідальність класу: Приймає HTTP-запити, пов’язані з фото працівників.

Функції, які надає клас:

- uploadPhoto() – завантаження фото працівника;
- getPhotoByEmployee() – отримання фото за працівником;
- deletePhoto() – видалення фото.

Назва: AdminPinController

Відповідальність класу: Надає доступ до операцій з PIN-кодами адміністраторів.

Функції, які надає клас:

- setPin() – встановлює PIN-код;
- verifyPin() – перевіряє правильність PIN-коду.

Назва: WorkTimeLogRepository

Відповідальність класу: Дає доступ до даних про журнал робочого часу.

Функції, які надає клас:

- findByEmployeeId() – знаходить записи журналу по працівнику;
- save() – зберігає журнал зміни;
- findAll() – повертає всі записи.

Назва: PhotoRepository

Відповідальність класу: Надає доступ до фото у базі даних.

Функції, які надає клас:

- findByEmployeeId() – знаходить фото працівника;
- deleteById() – видаляє фото за ID.

Назва: SalaryRepository

Відповідальність класу: Доступ до інформації про зарплати в базі.

Функції, які надає клас:

- findByEmployeeId() – отримує зарплату конкретного працівника;
- findAll() – повертає всі зарплати.

Назва: TaxSettingRepository

Відповідальність класу: Забезпечує збереження та читання податкових налаштувань.

Функції, які надає клас:

- findAll() – отримує всі податкові налаштування;
- save() – зберігає або оновлює параметри.

Назва: AdminPinRepository

Відповідальність класу: Забезпечує доступ до збережених PIN-кодів.

Функції, які надає клас:

- findById() – отримує PIN по ID;
- save() – зберігає PIN-код.

Назва: QRCodeController

Відповідальність класу: Обробляє запити на генерацію або перевірку QR-кодів.

Функції, які надає клас:

- generateQrForEmployee() – створює QR-код для працівника;
- validateQr() – перевіряє правильність коду.

Назва: QRCodeService (раніше згадано, але тепер детальніше)

Відповідальність класу: Логіка генерації та валідації QR-кодів.

Функції, які надає клас:

- generateForEmployee() – створює QR-код;
- parseQrData() – розпізнає закодовану інформацію;
- isValidQrCode() – перевіряє чи QR дійсний.

Назва: DateService

Відповідальність класу: Робота з датами для фільтрації чи розрахунків.

Функції, які надає клас:

- getCurrentDate() – повертає поточну дату;
- isWeekend() – перевіряє, чи день є вихідним.

Назва: TimeService

Відповідальність класу: Клас для роботи з часом та його форматуванням.

Функції, які надає клас:

- getCurrentTime() – поточний час;
- formatTime() – форматування часу у потрібний вигляд.

Назва: PortService

Відповідальність класу: Використовується для логіки взаємодії портів (можливо USB/камера).

Функції, які надає клас:

- listAvailablePorts() – повертає список доступних портів;
- connectToPort() – встановлює з'єднання.

Логічна модель даних

Для відображення структури бази даних, незалежно від обраної СКБД, було створено логічну модель даних.

Логічна модель даних представлено на рисунку 2.6.

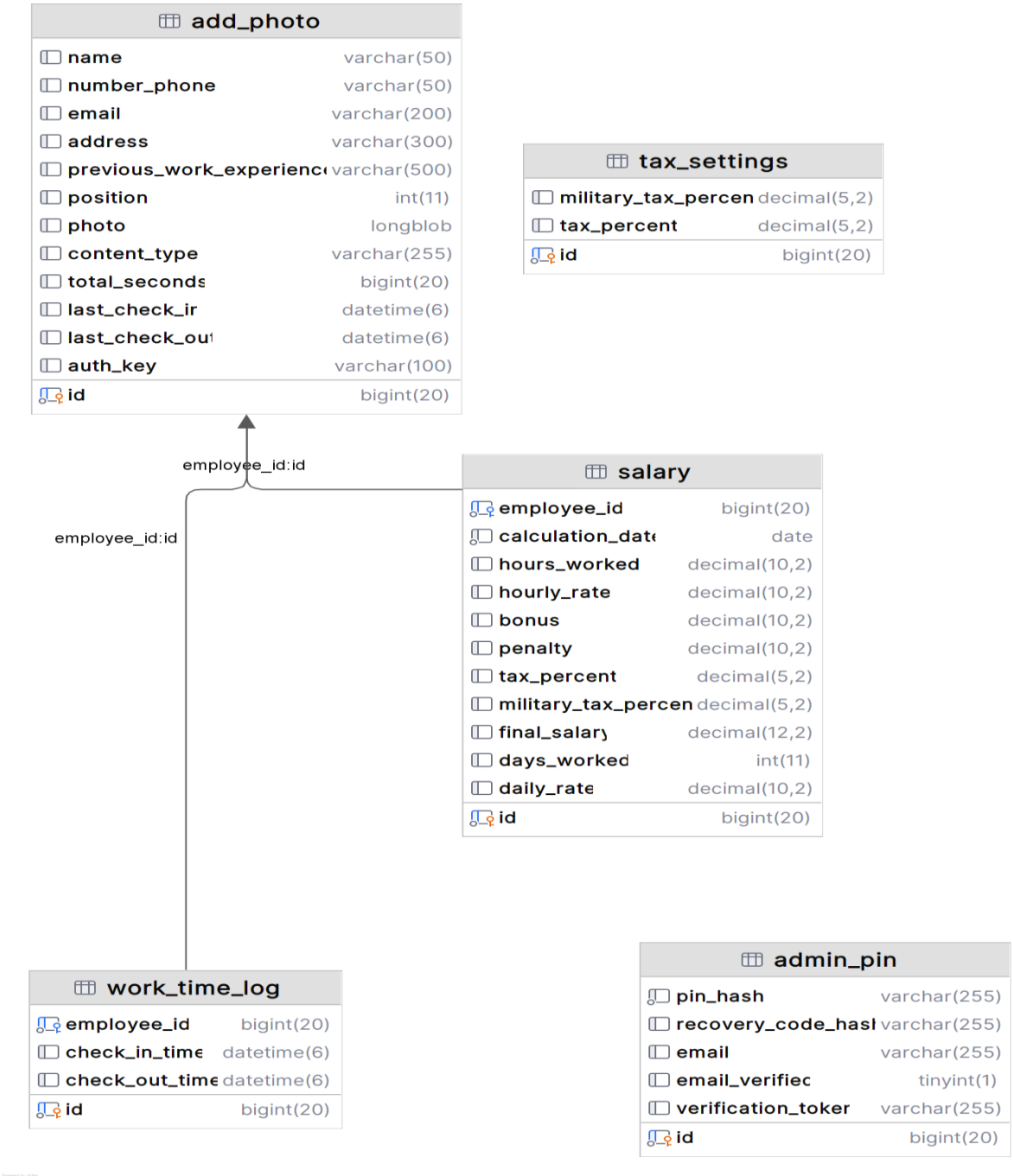

Рисунок 2.6 – Логічна модель даних

2.3.3 Робочий проєкт програмного забезпечення автоматизації нарахування заробітної плати

Вибір мови програмування та системи керування базою даних

У процесі розробки програмного забезпечення було використано низку сучасних технологій, що забезпечили ефективну реалізацію функціоналу як на стороні клієнта, так і на стороні сервера.

Для реалізації проєкту використано такі технології:

- Клієнтська частина реалізована з використанням HTML, CSS, JavaScript для створення зручного інтерфейсу користувача [9, 10]. Окремі компоненти, були реалізовані мовою Java.
- Серверна частина побудована з використанням Java, Framework Spring Boot та середовища Node.js, що забезпечує зручну організацію коду та обробку запитів [11]. У якості системи керування базою даних було обрано MariaDB — надійний та швидкий СКБД із підтримкою SQL [12].
- Мобільна частина програмного забезпечення створена за допомогою середовища Android Studio, а програмна логіка написана мовою Kotlin, яка є сучасною, безпечною й офіційно рекомендованою Google для розробки Android-додатків [13, 14].

Такий вибір інструментів дозволив забезпечити надійність системи та зручність для користувача.

Фізична модель даних

Фізичну модель бази даних представлені у вигляді таблиць 2.13-2.17.

Таблиця 2.13 – Структура таблиці add_photo

№	Назва поля	Ключ	Тип даних
1	id	PK	bigint(20)
2	name		varchar(50)
3	number_phone		varchar(50)
4	email		varchar(200)
5	address		varchar(300)
6	previous_work_experience		varchar(500)
7	position		int(11)
8	photo		longblob
9	content_type		varchar(255)
10	total_seconds		bigint(20)
11	last_check_ir		datetime(6)
12	last_check_out		datetime(6)
13	auth_key		varchar(100)

Таблиця 2.14 – Структура таблиці salary

№	Назва поля	Ключ	Тип даних
1	id	PK	bigint(20)
2	employee_id	FK	bigint(20)
3	calculation_date		date
4	hours_worked		decimal(10,2)
5	hourly_rate		decimal(10,2)
6	bonus		decimal(10,2)
7	penalty		decimal(10,2)
8	tax_percent		decimal(5,2)
9	military_tax_percent		decimal(5,2)

Кінець таблиці 2.14

10	final_salary		decimal(12,2)
11	days_worked		int(11)
12	daily_rate		decimal(10,2)

Таблиця 2.15 – Структура таблиці tax_settings

№	Назва поля	Ключ	Тип даних
1	id	PK	bigint(20)
2	military_tax_percen		decimal(5,2)
3	tax_percent		decimal(5,2)

Таблиця 2.16 – Структура таблиці work_time_log

№	Назва поля	Ключ	Тип даних
1	id	PK	bigint(20)
2	employee_id	FK	bigint(20)
3	check_in_time		datetime(6)
4	check_out_time		datetime(6)

Таблиця 2.17 – Структура таблиці admin_pin

№	Назва поля	Ключ	Тип даних
1	id	PK	bigint(20)
2	pin_hash		varchar(255)
3	recovery_code_hash		varchar(255)
4	email		varchar(255)
5	email_verified		tinyint(1)
6	verification_token		varchar(255)

Розроблено повнофункціональну версію програмного забезпечення, яка охоплює:

- повний облік персоналу та їхніх змін;
- точний підрахунок заробітної плати на основі фактичного часу роботи;
- можливість адміністрування та перегляду звітів;
- інтеграцію мобільного додатку для працівників із фіксацією часу через QR-код.

Було проведено тестування системи, яке показало її стабільну роботу.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

У результаті виконання кваліфікаційної роботи було розроблено повноцінне програмне забезпечення для нарахування заробітної плати в кав'ярні Lite Cafe. Реалізоване програмне забезпечення складається з вебінтерфейсу для адміністратора та мобільної частини програмного забезпечення для працівників.

Загальний обсяг програмного коду становить 4 868 рядків, з них:

- вебінтерфейсна частина – 3 791 рядка (HTML, CSS, JS, Node.js);
- мобільна частина – 1077 рядків (Kotlin).

Розмір скомпільованого коду:

- мобільна частина — 6.3 МБ;
- вебінтерфейсна частина — 14,7 МБ (Node.js, HTML, JS, CSS).

Програмне забезпечення відповідає функціональним та нефункціональним вимогам, зазначеним у технічному завданні.

Серед основних функцій, які реалізовані у програмному забезпеченні є:

- Авторизація адміністратора з використанням електронної пошти та PIN-коду;
- Забезпечення безпеки доступу — реалізовано резервне відновлення PIN-коду, зміну пароля та блокування програми;
- Управління персоналом — додавання, редагування, перегляд та видалення працівників із можливістю зберігання особистих даних;
- Фіксація робочого часу — працівники сканують QR-коди для відмітки приходу та виходу з роботи. Уся інформація зберігається в базі даних;

- Журнали відвідувань — адміністратор може переглядати календар змін та історію відвідувань кожного працівника;
- Розрахунок заробітної плати — автоматичне обчислення суми до виплати з урахуванням ставки, податків, премій і штрафів;
- Формування звітів — на основі введених даних програмного забезпечення генерує деталізовані звіти з результатами розрахунків;
- Телефонна частина програмного забезпечення — реалізований функціонал для працівників: сканування QR-коду, перегляд особистої інформації та статусу зміни.

Інтерфейси користувача реалізовано зручно та доступно. Було виконано тестування всіх модулів, яке підтвердило працездатність системи та її стабільну роботу.

Нижче представлено знімки екрана сторінок відділ кадрів та фінансовий відділ (рисунки 3.1–3.4).

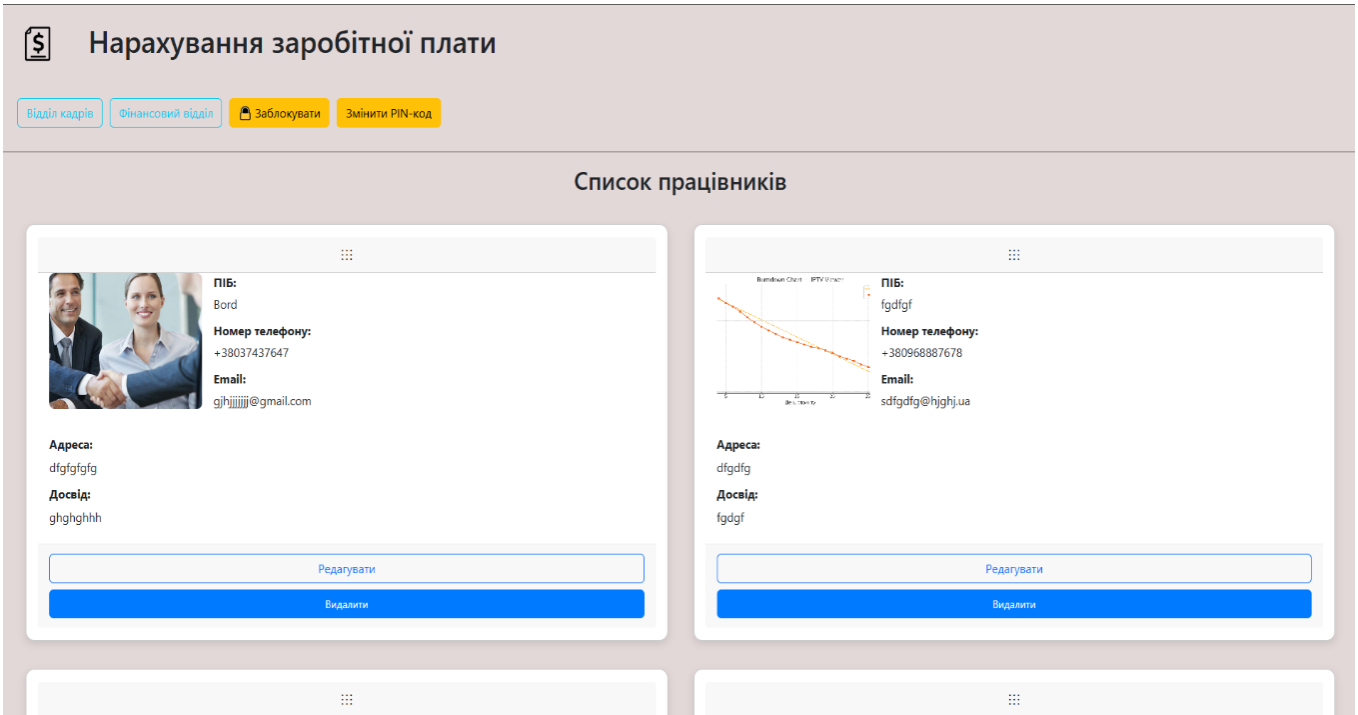

Рисунок 3.1 – Знімок екрана сторінки відділ кадрів (верхня частина)

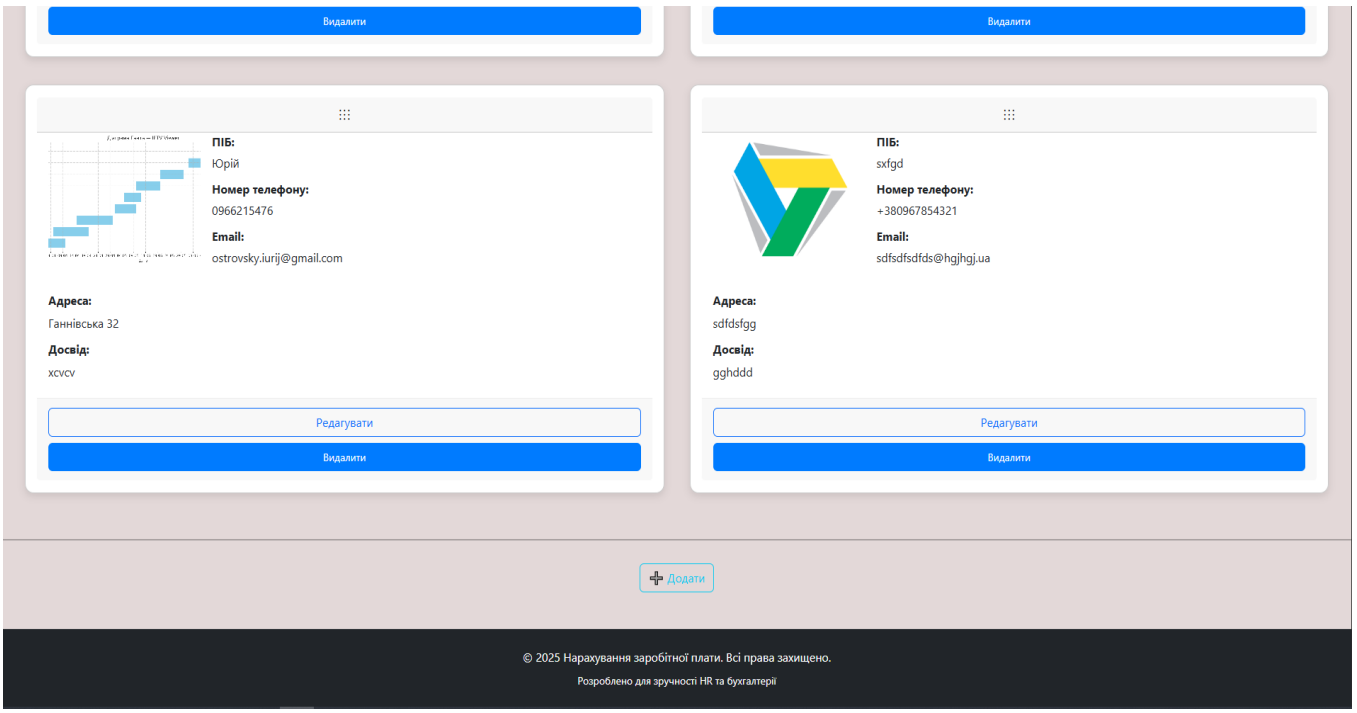

Рисунок 3.2 – Знімок екрана сторінки відділ кадрів (нижня частина)

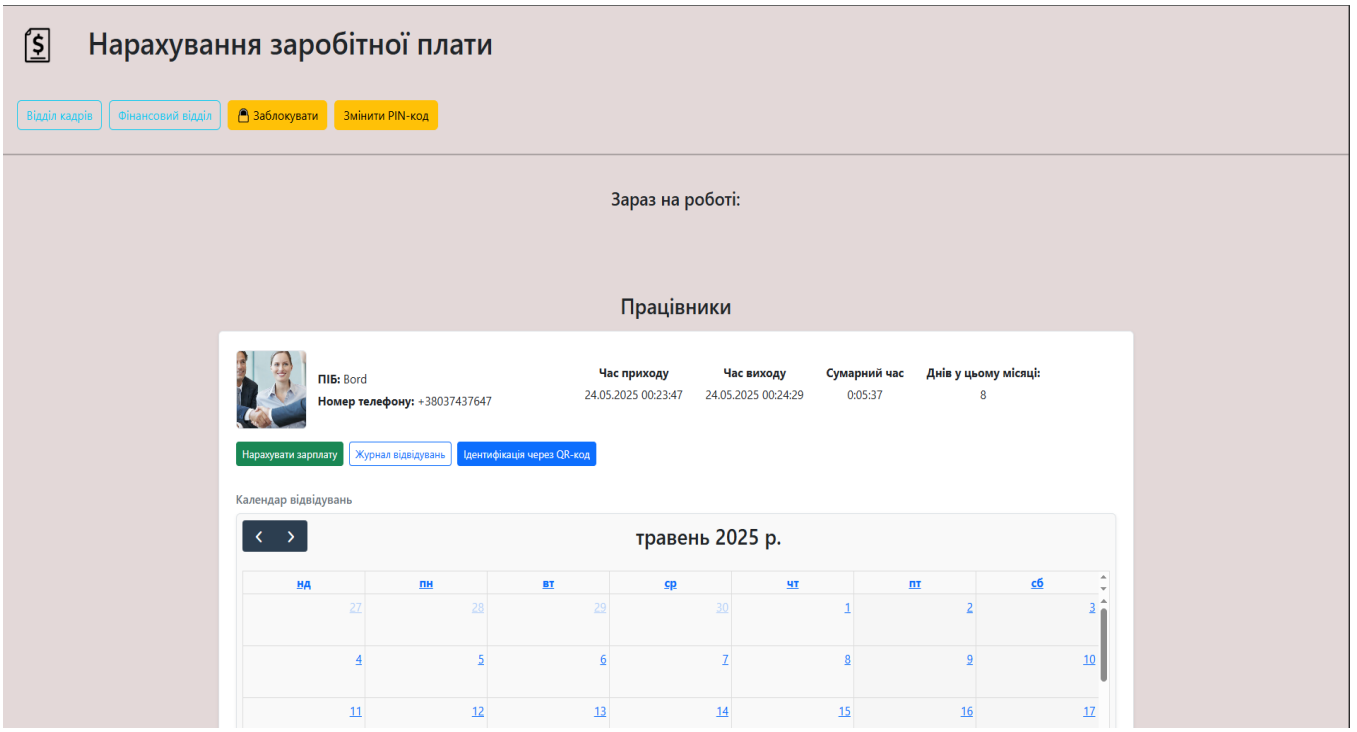

Рисунок 3.3 – Знімок екрана сторінки фінансовий відділ (верхня частина)

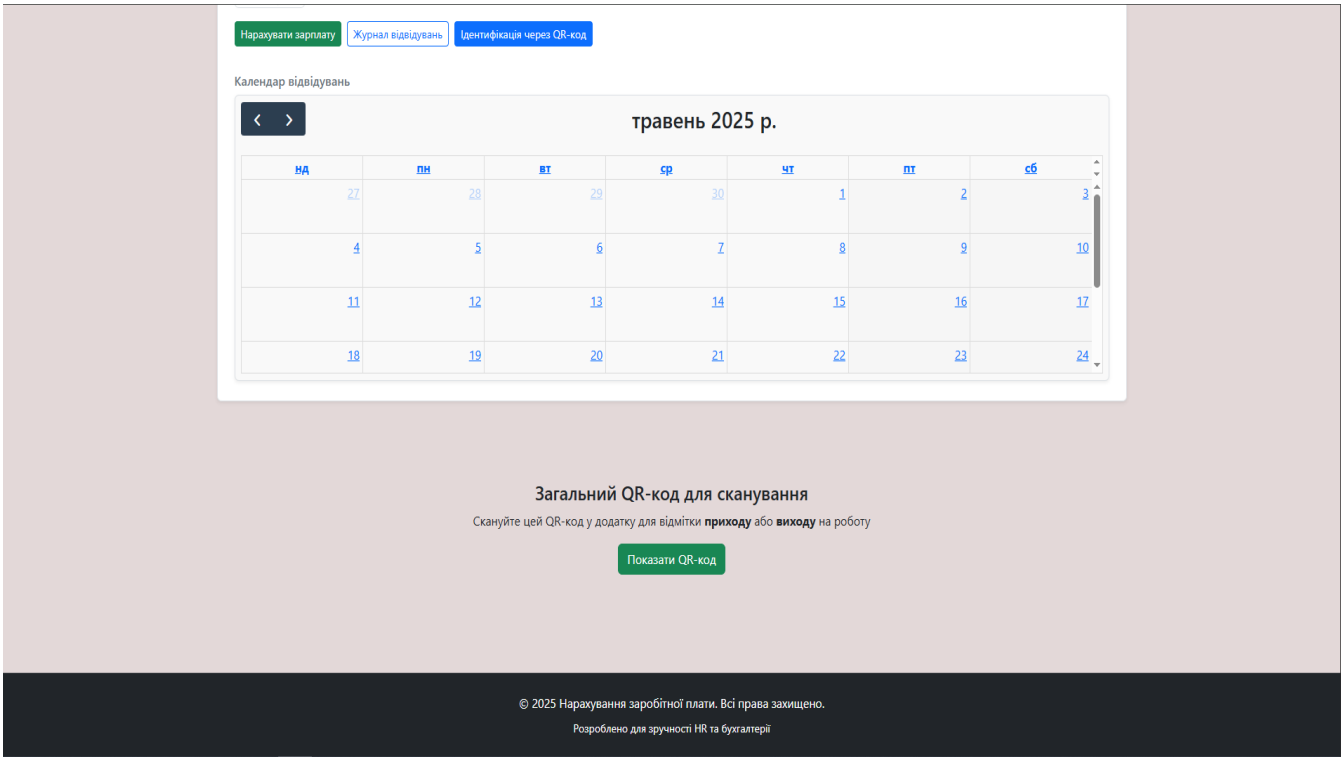

Рисунок 3.4 – Знімок екрана сторінки фінансовий відділ (нижня частина)

4 ОХОРОНА ПРАЦІ

Під час розробки програмного забезпечення для автоматизації нарахування заробітної плати для кав’ярні Lite Cafe, також було враховано питання охорони праці, зокрема умови роботи персоналу, що працює з комп’ютерною технікою.

Розробка відбувалась із урахуванням того, що програмне забезпечення буде використовуватись адміністратором в приміщенні кав’ярні. Тому при проєктуванні та впровадженні рішення, важливо створити безпечні та зручні умови для роботи з комп’ютером.

4.1 Умови для працівників, які використовують програмне забезпечення

Для користувачів програми повинно бути організоване робоче місце: стіл, крісло, комп’ютер або ноутбук, обов’язково розташованим на оптимальній відстані, приблизно 60–70 см, від очей.

Освітлення в приміщенні має бути достатнім. Важливо також забезпечити регулярне провітрювання для підтримання комфортного мікроклімату. Приміщення повинно регулярно провітрюватися.

4.2 Безпека під час роботи з технікою

Всі електричні прилади, які використовуються для роботи з програмою, мають бути технічно справними. Забороняється підключення обладнання до несправних розеток чи використання пошкоджених кабелів.

Для запобігання перегріванню пристроїв необхідно уникати перекриття вентиляційних отворів комп’ютера чи ноутбука.

4.3 Протипожежна безпека

Приміщення Lite Cafe має бути обладнане засобами пожежогасіння – як мінімум, вогнегасником. Працівники повинні бути ознайомлені з інструкцією щодо дій у випадку пожежі. Не допускається залишати ввімкнену техніку без нагляду після завершення роботи.

4.4 Перерви та режим роботи

Щоб уникнути перенавантаження, потрібно обов’язково організовувати короткі перерви під час роботи за комп’ютером, наприклад, 5–10 хвилин кожну годину. Це знижує навантаження на зір, а також допомагає підтримувати загальне самопочуття працівника.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було вирішено практичну задачу інженерії програмного забезпечення: розроблено програмне забезпечення для автоматизації нарахування заробітної плати в кав’ярні Lite Cafe.

Було проведено аналіз практичної задачі, яка вимагає автоматизації обліку робочого часу та розрахунків заробітної плати в умовах змінного графіка.

Проведено аналіз аналогічного програмного забезпечення (Shifton, Paychex), що дозволило виявити їх недоліки для малих закладів.

Виконано постановку задачі з урахуванням специфіки кав’ярні, включаючи необхідність мобільної частини та розрахунку зарплат з податками.

Спроєктовано архітектуру системи на основі клієнт-серверної моделі з використанням принципів ООП і шаблону MVC.

Реалізовано основні модулі: фіксація робочого часу, розрахунок заробітної плати, мобільна частина із підтримкою сканування QR-кодів.

Проведено тестування системи, яке підтвердило її працездатність і відповідність визначеним вимогам.

Розроблена програмне забезпечення дозволяє враховувати змінний графік, автоматично розраховувати зарплату з урахуванням податків, надбавок і штрафів, вести облік присутності та формувати фінансову звітність.

Програмне забезпечення зменшує обсяг ручної роботи, мінімізує кількість помилок у розрахунках та підвищує зручність і швидкість обліку в кав’ярні Lite Cafe.

Отже, мету кваліфікаційної роботи досягнуто, а результати можуть бути використані для впровадження в діяльність кав'ярні Lite Cafe або подібних закладів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Програмна інженерія. — 10th ed. — Boston: Pearson, 2015. — 816 p.
2. Бажан О.І. Основи програмної інженерії. — К.: Ліра-К, 2021. — 320 с.
3. Shifton. Програмне забезпечення автоматизації графіків та обліку часу. URL: https://shifton.com/ru/
4. Paychex. Payroll Software Solutions. URL: https://www.paychex.eu/
5. ISO/IEC 25010:2011. Системна та програмна інженерія — Вимоги до якості систем та програмного забезпечення та оцінка (SQuaRE) — Моделі якості систем та програмного забезпечення.
6. UML Use Case Diagrams — Lucid chart Guide. URL:https://www.lucidchart.com/pages/uml-use-case-diagram
7. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Шаблони проєктування. Стандарти рішень об'єктно-орієнтованого програмування. — К.: Діалектика, 2002. — 352 с.
8. Java SE Documentation. URL: https://docs.oracle.com/javase/8/docs/
9. W3Schools. HTML, CSS, JavaScript Tutorial. URL:https://www.w3schools.com/
10. Mozilla Developer Network (MDN). https://developer.mozilla.org/
11. Node.js Documentation. URL:https://nodejs.org/en/docs
12. MariaDB Knowledge Base. URL:https://mariadb.com/kb/en/
13. Android Developers: Kotlin Language. URL:https://developer.android.com/kotlin
14. Kotlin Documentation. https://kotlinlang.org/docs/home.html

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ’ЯРНІ LITE CAFE

1 Вступ

1.1 Назва програмного забезпечення

Програмне забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe.

1.2 Сфера застосування

Програмне забезпечення призначене для використання в малих підприємствах громадського харчування, зокрема в кав’ярні Lite Cafe, з метою ведення обліку робочого часу персоналу та розрахунку заробітної плати.

2 Підстава для розробки програмного забезпечення

Розробка програмного забезпечення виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

3 Призначення розробки

3.1 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація процесу обліку відпрацьованих годин та нарахування заробітної плати з урахуванням змінного графіка, податків, штрафів, військових зборів та надбавок у вигляді премії.

3.2 Експлуатаційне призначення

Програмне забезпечення буде використовуватись адміністрацією кав’ярні для внутрішніх розрахунків оплати праці. Контролювання робочого часу та розрахунку заробітної плати

4 Технічні вимоги до програми

4.1 Вимоги до функціональних характеристик

4.1.1 Вимоги до переліку виконуваних функцій

Перелік виконуваних функцій програмного забезпечення

- Введення та редагування даних працівників;
- Облік відпрацьованих годин;
- Розрахунок заробітної плати з урахуванням податків, премій та штрафів;

- Можливість використовувати телефонну частину програмного забезпечення, призначений для працівників
- Безпека, блокування програмного забезпечення для адміністратора на випадок відійти від робочого місця і не втручання інших працівників до програмного забезпечення

4.1.2 Вимоги до організації вхідних та вихідних даних

1) Введення та редагування даних працівників
   Вхідні дані: ПІБ, номер телефону, фото, електрона адреса, місце проживання, попередній досвід роботи.
   Вихідні дані: Збережені та оновлені записи працівників у базі.
2) Облік відпрацьованих годин
   Вхідні дані: Час входу та виходу, сканування QR-коду для ідентифікування у додатку.
   Вихідні дані: Таблиця з відпрацьованими годинами, загальний підсумок за зміну/місяць.
3) Розрахунок заробітної плати з урахуванням податків, премій та штрафів
   Вхідні дані: Дані обліку часу, ставка, коефіцієнти податків, розміри премій і штрафів.
   Вихідні дані: Розрахована сума до виплати.
4) Можливість використовувати телефонну частину програмного забезпечення, призначений для працівників
   Вхідні дані: Ідентифікаційний ключ або QR-код для ідентифікації.
   Вихідні дані: Інтерфейс особисті дані працівника для перегляду.
5) Безпека, блокування додатку для адміністратора

Вхідні дані: PIN-код, електрона адреса адміністратора, резервний PIN-код.

Вихідні дані: Заблокований доступ, розблокований доступ.

4.2 Вимоги до надійності

Програма забезпечує коректну роботу при збоїв живлення або закритті. Надійно зберігає дані та уникає втрати інформації.

4.3 Умови експлуатації

4.3.1 Кліматичні умови експлуатації

Робота в приміщенні з температурою від +10 до +35 °C.

4.3.2 Вимоги до чисельності та кваліфікації користувачів

Програма призначена для роботи користувачем з базовими навичками володіння ПК або ноутбуком і телефоном

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Вимоги до користувацького обладнання

- ПК або ноутбук з ОС Windows 10 або вище;
- Оперативна пам'ять: від 2 ГБ;
- Місце на диску: від 500 МБ

4.4.2 Вимоги до серверного обладнання

У випадку мережевої версії – локальний сервер або хостинг з підтримкою СУБД (наприклад, MySQL/PostgreSQL).

4.5 Вимоги до інформаційної та програмної сумісності

Клієнт повинен мати в наявності наступні програмні засоби:

- Клієнт: браузер Google Chrome або Mozilla Firefox, або інтерфейс програма Windows та телефонну частину програмного забезпечення.

Сервер повинен відповідати наступним вимогам до програмної сумісності:

- Сервер: ОС Linux або Windows Server, підтримка HTML/Java та СУБД.

4.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки відсутні.

4.7 Вимоги до транспортування та зберігання

Вимоги до маркування та упаковки відсутні.

4.8 Спеціальні вимоги

Спеціальні вимоги відсутні.

5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- текст програми;
- керівництво користувача;
- опис програми;
- програму і методику випробувань.

6 Техніко-економічні показники

У кваліфікаційній роботі техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Етапи розробки програмного забезпечення наведено в таблиці А.1.

Таблиця А.1 – Етапи розробки програмного забезпечення

Номер	Назва етапів роботи	Терміни виконання
1.	Аналіз практичної задачі інженерії ПЗ	05.04.2025
2.	Аналіз вимог до ПЗ	12.04.2025
3.	Розробка архітектури ПЗ	19.04.2025
4.	Розробка проєкту ПЗ	02.05.2025
5.	Реалізація та тестування ПЗ	15.05.2025

8 Порядок контролю та прийомки

Контроль здійснюється керівником кваліфікаційної роботи на кожному етапі розробки програмного забезпечення. Приймання здійснюється за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

ДОДАТОК Б – ТЕКСТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

entities/Photo.java

```
package com.example.final20.entities;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDateTime;
import java.util.Base64;
import java.util.List;
import java.util.Random;

@Getter
@Setter
@Entity
@Table(name = "add photo")
public class Photo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name", length = 50)
    private String name;

    @Column(name = "number_phone", length = 50)
    private String number_phone;

    @Column(name = "email", length = 200)
    private String email;

    @Column(name = "address", length = 300)
    private String address;

    @Column(name = "previous_work_experience", length = 500)
    private String previous_work_experience;

    @Column(name = "position")
    private Integer position;

    @Column(name = "auth_key", length = 100)
    private String authKey;

    public String generateRandomKey() {
        int length = 20;
        String chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
```

```
        StringBuilder sb = new StringBuilder();
        Random random = new Random();
        for (int i = 0; i < length; i++) {
            sb.append(chars.charAt(random.nextInt(chars.length())));
        }
        return sb.toString();
    }

    @PrePersist
    public void prePersist() {
        if (this.authKey == null || this.authKey.isEmpty()) {
            this.authKey = generateRandomKey();
        }
    }

    private Long totalSeconds;

    @Lob
    @Column(name = "photo", columnDefinition = "LONGBLOB")
    private byte[] photo;

    @Column(name = "content_type")
    private String contentType;


    public String getPhotoBase64() {
        return this.photo != null ?
Base64.getEncoder().encodeToString(this.photo) : null;
    }

    private LocalDateTime lastCheckIn;
    private LocalDateTime lastCheckOut;

    @OneToMany(mappedBy = "employee", cascade = CascadeType.ALL)
    private List<Salary> salaries;

    @Transient
    private Integer daysWorkedThisMonth;
}
```

entities/WorkTimeLog.java

```
package com.example.final20.entities;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDateTime;

@Getter
@Setter
@Entity
@Table(name = "work_time_log")
public class WorkTimeLog {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @ManyToOne
    @JoinColumn(name = "employee_id", nullable = false)
    private Photo employee;

    @Column(name = "check_in_time")
    private LocalDateTime checkInTime;

    @Column(name = "check_out_time")
    private LocalDateTime checkOutTime;
}
```

entities/Salary.java

```
package com.example.final20.entities;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.math.BigDecimal;
import java.time.LocalDate;

@Entity
@Table(name = "salary")
@Getter
@Setter
public class Salary {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "employee_id", nullable = false)
    private Photo employee;

    @Column(name = "calculation date", nullable = false)
    private LocalDate calculationDate;

    @Column(name = "hours_worked", precision = 10, scale = 2)
    private BigDecimal hoursWorked;

    @Column(name = "hourly_rate", precision = 10, scale = 2)
    private BigDecimal hourlyRate;

    @Column(name = "bonus", precision = 10, scale = 2)
    private BigDecimal bonus;

    @Column(name = "penalty", precision = 10, scale = 2)
    private BigDecimal penalty;

    @Column(name = "tax_percent", precision = 5, scale = 2)
    private BigDecimal taxPercent;

    @Column(name = "military_tax_percent", precision = 5, scale = 2)
    private BigDecimal militaryTaxPercent;
```

```
    @Column(name = "final_salary", precision = 12, scale = 2)
    private BigDecimal finalSalary;

    @Column(name = "days_worked")
    private Integer daysWorked;

    @Column(name = "daily_rate", precision = 10, scale = 2)
    private BigDecimal dailyRate;
}
```

entities/TaxSettings.java

```
package com.example.final20.entities;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import org.springframework.data.annotation.Id;

import java.math.BigDecimal;

@Entity
@Getter
@Setter
@Table(name = "tax_settings")
public class TaxSettings {

    @jakarta.persistence.Id
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "tax_percent", precision = 5, scale = 2)
    private BigDecimal taxPercent;

    @Column(name = "military_tax_percent", precision = 5, scale = 2)
    private BigDecimal militaryTaxPercent;
}
```

entities/AdminPin.java

```
package com.example.final20.entities;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
public class AdminPin {
    @Id
```

```
    private Long id = 1L;

    @Column(nullable = false)
    private String pinHash;

    @Column
    private String recoveryCodeHash;

    @Column
    private String email;

    @Column
    private boolean emailVerified;

    public Boolean getEmailVerified() {
        return emailVerified;
    }

    public void setEmailVerified(Boolean emailVerified) {
        this.emailVerified = emailVerified;
    }

    @Column
    private String verificationToken;
}
```

controllers/AdminPinController.java

```
package com.example.final20.controllers;

import com.example.final20.entities.AdminPin;
import com.example.final20.service.AdminPinService;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.UUID;

@Controller
@RequestMapping("/admin")
public class AdminPinController {

    private final AdminPinService adminPinService;

    public AdminPinController(AdminPinService adminPinService) {
        this.adminPinService = adminPinService;
    }

    @GetMapping("/unlock")
    public String showUnlockPage(HttpSession session, Model model,
                                 @RequestParam(required = false) String error,
                                 @RequestParam(required = false) String
resetSuccess,
                                 @ModelAttribute("message") String message,
```

```
                                @ModelAttribute("error") String flashError) {
        boolean pinExists = adminPinService.pinExists();
        model.addAttribute("pinExists", pinExists);

        if (error != null) {
            model.addAttribute("error", "Неправильний PIN-код. Спробуйте ще
раз.");
        }

        if (resetSuccess != null) {
            model.addAttribute("message", "PIN успішно оновлено. Увійдіть з
новим PIN.");
        }

        if (message != null && !message.isEmpty()) {
            model.addAttribute("message", message);
        }
        if (flashError != null && !flashError.isEmpty()) {
            model.addAttribute("error", flashError);
        }

        return "admin/unlock";
    }


    @PostMapping("/unlock")
    public String processUnlock(@RequestParam String pin, HttpSession session) {
        if (adminPinService.verifyPin(pin)) {
            session.setAttribute("adminUnlocked", true);
            return "redirect:/";
        } else {
            return "redirect:/admin/unlock?error=true";
        }
    }

    @PostMapping("/create-pin")
    public String createPin(@RequestParam String email,
                            @RequestParam String newPin,
                            @RequestParam String recoveryCode,
                            RedirectAttributes redirectAttributes) {
        if (adminPinService.pinExists()) {
            redirectAttributes.addFlashAttribute("error", "PIN вже створено.");
            return "redirect:/admin/unlock";
        }

        adminPinService.savePin(email, newPin, recoveryCode);

        try {
            adminPinService.sendEmailVerification(email);
            redirectAttributes.addFlashAttribute("message", "PIN створено. Лист
для підтвердження надіслано на email.");
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("error", "Помилка надсилання
листа підтвердження: " + e.getMessage());
        }

        return "redirect:/admin/unlock";
    }
```

```
    @GetMapping("/change-pin")
    public String showChangePinForm() {
        return "admin/change-pin";
    }

    @PostMapping("/change-pin")
    public String changePin(@RequestParam String oldPin, @RequestParam String
newPin) {
        if (adminPinService.verifyPin(oldPin)) {
            adminPinService.updatePin(newPin);
            return "redirect:/admin/dashboard?pinChanged";
        } else {
            return "redirect:/admin/change-pin?error";
        }
    }

    @GetMapping("/forgot-pin")
    public String forgotPinForm() {
        return "admin/forgot-pin";
    }

    @PostMapping("/reset-pin")
    public String resetPin(@RequestParam String recoveryCode, @RequestParam
String newPin, HttpSession session) {
        if (adminPinService.verifyRecoveryCode(recoveryCode)) {
            adminPinService.updatePin(newPin);
            session.setAttribute("adminUnlocked", true);
            return "redirect:/admin/dashboard?reset=true";
        }
        return "redirect:/admin/forgot-pin?error";
    }

    @GetMapping("/dashboard")
    public String adminDashboard() {
        return "admin/dashboard";
    }

    @GetMapping("/reset-pin-form")
    public String showResetPinForm(@RequestParam(required = false) String token,
Model model) {
        AdminPin pin = adminPinService.getPin();
        if (token == null || pin == null ||
!token.equals(pin.getVerificationToken())) {
            return "redirect:/admin/forgot-pin?error=invalidToken";
        }
        model.addAttribute("token", token);
        return "admin/reset-pin-form";
    }

    @PostMapping("/reset-pin-email")
    public String processResetFromEmail(@RequestParam String newPin,
@RequestParam String token) {
        AdminPin pin = adminPinService.getPin();
        if (pin != null && token.equals(pin.getVerificationToken())) {
            adminPinService.updatePin(newPin);
            pin.setVerificationToken(null);
            adminPinService.save(pin);
            return "redirect:/admin/unlock?resetSuccess";
        }
```

```
            return "redirect:/admin/unlock?error=invalidToken";
        }

        @GetMapping("/change-recovery")
        public String showChangeRecoveryForm() {
            return "admin/change-recovery";
        }

        @PostMapping("/change-recovery")
        public String changeRecoveryCode(@RequestParam String oldRecoveryCode,
                                         @RequestParam String newRecoveryCode) {
            if (adminPinService.verifyRecoveryCode(oldRecoveryCode)) {
                adminPinService.updateRecoveryCode(newRecoveryCode);
                return "redirect:/admin/dashboard?recoveryChanged";
            } else {
                return "redirect:/admin/change-recovery?error";
            }
        }

        @GetMapping("/change-recovery-code-form")
        public String showChangeRecoveryCodeForm() {
            return "admin/change-recovery";
        }

        @GetMapping("/lock")
        public String lockAdmin(HttpSession session) {
            session.invalidate();
            return "redirect:/admin/lock-complete";
        }

        @GetMapping("/lock-complete")
        public String lockRedirect() {
            return "redirect:/admin/unlock";
        }

        @PostMapping("/send-verification")
        public String sendVerification(@RequestParam("email") String email,
    RedirectAttributes redirectAttributes) {
            adminPinService.sendEmailVerification(email);
            redirectAttributes.addFlashAttribute("message", "Посилання для
    підтвердження надіслано на email");
            return "redirect:/admin/unlock";
        }


        @GetMapping("/request-verification")
        public String requestEmailVerification(RedirectAttributes
    redirectAttributes) {
            AdminPin pin = adminPinService.getPin();
            if (pin == null || pin.getEmail() == null) {
                redirectAttributes.addFlashAttribute("error", "Email не
    встановлено.");
                return "redirect:/admin/dashboard";
            }

            adminPinService.sendEmailVerification(pin.getEmail());
            redirectAttributes.addFlashAttribute("message", "Лист для підтвердження
    надіслано.");
            return "redirect:/admin/dashboard";
```

```
    }

    @GetMapping("/verify-email")
    public String verifyEmail(@RequestParam("token") String token,
RedirectAttributes redirectAttributes) {
        AdminPin pin = adminPinService.getPin();
        if (pin != null && token.equals(pin.getVerificationToken())) {
            pin.setEmailVerified(true);
            pin.setVerificationToken(null);
            adminPinService.save(pin);
            redirectAttributes.addFlashAttribute("message", "Email успішно
підтверджено!");
        } else {
            redirectAttributes.addFlashAttribute("error", "Невірний або
протермінований токен.");
        }
        return "redirect:/admin/dashboard";
    }

    @PostMapping("/send-reset-link")
    public String sendResetLink(HttpServletRequest request) {
        AdminPin pin = adminPinService.getPin();

        if (pin == null || pin.getEmail() == null ||
!Boolean.TRUE.equals(pin.getEmailVerified())) {
            return "redirect:/admin/forgot-pin?error=emailNotVerified";
        }

        String token = adminPinService.generateVerificationToken();
        pin.setVerificationToken(token);
        adminPinService.save(pin);

        String resetLink = request.getScheme() + "://" + request.getServerName()
+ ":" + request.getServerPort()
                + "/admin/reset-pin-form?token=" + token;

        adminPinService.sendResetPinEmail(pin.getEmail(), resetLink);
        return "redirect:/admin/forgot-pin?sent";
    }

    @GetMapping("/")
    public String index(HttpSession session) {
        if (session.getAttribute("adminUnlocked") == null) {
            return "redirect:/admin/unlock";
        }
        return "index";
    }
}
```

controllers/AttendanceController.java

```
package com.example.final20.controllers;

import com.example.final20.dao.WorkTimeLogRepository;
import com.example.final20.dto.*;
import com.example.final20.entities.Photo;
import com.example.final20.entities.WorkTimeLog;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
```

```
import org.springframework.http.*;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.*;

@Controller
@RequestMapping("/attendance")
public class AttendanceController {

    private final JdbcTemplate jdbcTemplate;
    private final DateService dateService;
    private final QrCodeService qrCodeService;
    private final WorkTimeService workTimeService;
    private final WorkTimeLogRepository workTimeLogRepository;

    public AttendanceController(JdbcTemplate jdbcTemplate, DateService
dateService, QrCodeService qrCodeService,
                                WorkTimeService workTimeService,
WorkTimeLogRepository workTimeLogRepository) {
        this.jdbcTemplate = jdbcTemplate;
        this.dateService = dateService;
        this.qrCodeService = qrCodeService;
        this.workTimeService = workTimeService;
        this.workTimeLogRepository = workTimeLogRepository;
    }

    @PostMapping("/check-in/{employeeId}")
    public ResponseEntity<String> checkIn(@PathVariable Long employeeId) {
        String sql = "INSERT INTO work_time_log (employee_id, check_in_time)
VALUES (?, NOW())";
        jdbcTemplate.update(sql, employeeId);
        return ResponseEntity.ok("Прихід зафіксовано");
    }

    @PostMapping("/check-out/{employeeId}")
    public ResponseEntity<String> checkOut(@PathVariable Long employeeId) {
        String sql = "UPDATE work_time_log SET check_out_time = NOW() " +
                "WHERE employee_id = ? AND check_out_time IS NULL ORDER BY
check_in_time DESC LIMIT 1";
        int updated = jdbcTemplate.update(sql, employeeId);
        if (updated == 0) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Не
знайдено активного запису про прихід");
        }
        return ResponseEntity.ok("Вихід зафіксовано");
    }

    @GetMapping("/qr-post/{action}/{employeeId}")
    public String showQrPost(@PathVariable String action,
                             @PathVariable Long employeeId,
                             Model model) {
        model.addAttribute("employeeId", employeeId);
        model.addAttribute("action", action);
```

```
        return "qr-post";
    }

    @GetMapping("/history/{employeeId}")
    public String employeeAttendanceHistory(@PathVariable Long employeeId, Model
model) {
        String sql = "SELECT w.id, w.check_in_time, w.check_out_time, p.name " +
                "FROM work_time_log w JOIN add_photo p ON w.employee_id = p.id "
+
                "WHERE w.employee_id = ? ORDER BY w.check_in_time DESC";
        List<AttendanceRecord> history = jdbcTemplate.query(sql, new
Object[]{employeeId}, (rs, rowNum) -> {
            AttendanceRecord record = new AttendanceRecord();
            record.setId(rs.getLong("id"));
            record.setName(rs.getString("name"));

record.setCheck_in_time(rs.getTimestamp("check_in_time").toLocalDateTime());
            record.setCheck_out_time(rs.getTimestamp("check_out_time") != null ?
                    rs.getTimestamp("check_out_time").toLocalDateTime() : null);
            return record;
        });

        model.addAttribute("history", history);
        model.addAttribute("hasHistory", !history.isEmpty());
        model.addAttribute("dateService", dateService);
        return "attendance-history";
    }

    @GetMapping("/calendar-data/{employeeId}")
    @ResponseBody
    public List<Map<String, Object>> getCalendarData(@PathVariable Long
employeeId) {
        String sql = """
        SELECT w.id, w.check_in_time, w.check_out_time, p.name
        FROM work_time_log w
        JOIN add_photo p ON w.employee_id = p.id
        WHERE w.employee_id = ?
    """;

        return jdbcTemplate.query(sql, new Object[]{employeeId}, (rs, rowNum) ->
{
            Map<String, Object> event = new HashMap<>();

            String name = rs.getString("name");
            LocalDateTime checkIn =
rs.getTimestamp("check_in_time").toLocalDateTime();
            Timestamp checkOutTs = rs.getTimestamp("check_out_time");
            LocalDateTime checkOut = checkOutTs != null
                    ? checkOutTs.toLocalDateTime()
                    : checkIn.plusHours(1);

            event.put("title", name);
            event.put("start", checkIn.toString());
            event.put("end", checkOut.toString());

            if (checkOutTs != null) {
                event.put("color", "#28a745");
            } else {
                event.put("color", "#dc3545");
```

```
            }

            return event;
        });
    }

    @PostMapping("/delete/{id}")
    public String deleteAttendance(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
        WorkTimeLog log = workTimeLogRepository.findById(id).orElseThrow();
        Long employeeId = log.getEmployee().getId();

        workTimeLogRepository.deleteById(id);

        redirectAttributes.addFlashAttribute("message", "Запис видалено");
        return "redirect:/attendance/history/" + employeeId;
    }


    @GetMapping
    public String attendancePage(HttpSession session, HttpServletResponse
response, Model model) {
        if (session.getAttribute("adminUnlocked") == null) {
            return "redirect:/admin/unlock";
        }

        response.setHeader("Cache-Control", "no-cache, no-store, must-
revalidate");
        response.setHeader("Pragma", "no-cache");
        response.setDateHeader("Expires", 0);

        String photoSql = "SELECT id, name, number_phone, photo, auth_key FROM
add_photo ORDER BY name";
        List<Photo> photos = jdbcTemplate.query(photoSql, (rs, rowNum) -> {
            Photo p = new Photo();
            p.setId(rs.getLong("id"));
            p.setName(rs.getString("name"));
            p.setNumber_phone(rs.getString("number_phone"));
            p.setPhoto(rs.getBytes("photo"));
            p.setAuthKey(rs.getString("auth_key"));
            return p;
        });

        List<Map<String, Object>> employees = jdbcTemplate.queryForList(
                "SELECT id, name, number_phone, photo FROM add_photo ORDER BY
name"
        );
        Map<Long, Long> totalTimes = workTimeService.getTotalWorkTimes();
        for (Photo p : photos) {
            Long seconds = totalTimes.getOrDefault(p.getId(), 0L);
            p.setTotalSeconds(seconds);
        }

        String timeSql = """
            SELECT employee_id,
                   MAX(CASE WHEN check_in_time IS NOT NULL THEN check_in_time END)
AS last_check_in,
                   MAX(CASE WHEN check_out_time IS NOT NULL THEN check_out_time END)
AS last_check_out
```

```
        FROM work_time_log
        GROUP BY employee_id
    """;
        Map<Long, LocalDateTime> lastCheckIn = new HashMap<>();
        Map<Long, LocalDateTime> lastCheckOut = new HashMap<>();

        jdbcTemplate.query(timeSql, rs -> {
            long id = rs.getLong("employee_id");
            Timestamp checkIn = rs.getTimestamp("last_check_in");
            Timestamp checkOut = rs.getTimestamp("last_check_out");
            if (checkIn != null) lastCheckIn.put(id, checkIn.toLocalDateTime());
            if (checkOut != null) lastCheckOut.put(id,
checkOut.toLocalDateTime());
        });

        for (Photo p : photos) {
            Long id = p.getId();
            p.setTotalSeconds(totalTimes.getOrDefault(id, 0L));
            p.setLastCheckIn(lastCheckIn.getOrDefault(id, null));
            p.setLastCheckOut(lastCheckOut.getOrDefault(id, null));

            LocalDateTime startOfMonth =
LocalDateTime.now().withDayOfMonth(1).toLocalDate().atStartOfDay();
            LocalDateTime now = LocalDateTime.now();

            String workLogSql = """
        SELECT COUNT(DISTINCT DATE(check in time)) AS days
        FROM work_time_log
        WHERE employee_id = ? AND check_in_time BETWEEN ? AND ?
    """;

            Integer daysWorked = jdbcTemplate.queryForObject(
                    workLogSql,
                    new Object[]{id, Timestamp.valueOf(startOfMonth),
Timestamp.valueOf(now)},
                    Integer.class
            );

            p.setDaysWorkedThisMonth(daysWorked);
        }

        String onWorkSql = "SELECT DISTINCT p.name FROM work_time_log w " +
                "JOIN add_photo p ON w.employee_id = p.id " +
                "WHERE w.check_out_time IS NULL";
        List<String> onWork = jdbcTemplate.query(onWorkSql, (rs, rowNum) ->
rs.getString("name"));

        model.addAttribute("photos", photos);
        model.addAttribute("onWork", onWork);
        model.addAttribute("employees", employees);
        model.addAttribute("totalTimes", totalTimes);
        model.addAttribute("dateService", dateService);
        model.addAttribute("qrCodeService", qrCodeService);

        return "attendance";
    }
}
```

controllers/PhotoController.java

```
package com.example.final20.controllers;

import com.example.final20.dao.WorkTimeLogRepository;
import com.example.final20.service.PdfExportService;
import com.example.final20.entities.Photo;
import com.example.final20.dao.PhotoRepository;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import org.apache.tomcat.util.http.fileupload.ByteArrayOutputStream;
import org.springframework.data.domain.Sort;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Objects;

@Controller
public class PhotoController {

    private final PdfExportService pdfExportService;
    private final WorkTimeLogRepository workTimeLogRepository;
    private final PhotoRepository photoRepository;

    public PhotoController(PhotoRepository photoRepository, PdfExportService
pdfExportService, WorkTimeLogRepository workTimeLogRepository) {
        this.photoRepository = photoRepository;
        this.pdfExportService = pdfExportService;
        this.workTimeLogRepository = workTimeLogRepository;
    }

    @GetMapping("/photo/qrcode/{id}")
    @ResponseBody
    public ResponseEntity<byte[]> generateQrCode(@PathVariable Long id) {
        Photo photo = photoRepository.findById(id).orElse(null);
        if (photo == null || photo.getAuthKey() == null) {
            return ResponseEntity.notFound().build();
        }

        try {
            String loginText = "AUTH:" + photo.getAuthKey();
            QRCodeWriter qrCodeWriter = new QRCodeWriter();
            BitMatrix bitMatrix = qrCodeWriter.encode(loginText,
```

```
BarcodeFormat.QR_CODE, 300, 300);

            ByteArrayOutputStream pngOutputStream = new ByteArrayOutputStream();
            MatrixToImageWriter.writeToStream(bitMatrix, "PNG",
pngOutputStream);
            byte[] pngData = pngOutputStream.toByteArray();

            return ResponseEntity.ok()
                    .contentType(MediaType.IMAGE_PNG)
                    .body(pngData);
        } catch (WriterException | IOException e) {
            e.printStackTrace();
            return ResponseEntity.internalServerError().build();
        }
    }

    @GetMapping("/api/check-auth-status")
    @ResponseBody
    public Map<String, Object> checkAuthStatus(HttpSession session) {
        Map<String, Object> response = new HashMap<>();
        Photo photo = (Photo) session.getAttribute("authenticatedUser");

        if (photo != null) {
            response.put("success", true);
        } else {
            response.put("error", true);
        }

        return response;
    }

    @GetMapping("/mobile-login")
    public String mobileLogin(@RequestParam("key") String key, HttpSession
session, Model model) {
        Photo photo = photoRepository.findByAuthKey(key);
        if (photo == null) {
            return "error";
        }
        session.setAttribute("authenticatedUser", photo);
        model.addAttribute("photo", photo);
        return "mobile-dashboard";
    }

    @PostMapping("/photo/upload")
    public String uploadPhoto(@RequestParam("photo") MultipartFile file,
                              @RequestParam("name") String name,
@RequestParam("number_phone") String number_phone,
                              @RequestParam("email") String email,
@RequestParam("address") String address,
                              @RequestParam("previous_work_experience") String
previous_work_experience) {
        try {
            if
(!Objects.requireNonNull(file.getContentType()).startsWith("image/")) {
                throw new IllegalArgumentException("Файл не є зображенням.");
            }

            Photo photo = new Photo();
            photo.setName(name);
```

```
            photo.setNumber_phone(number_phone);
            photo.setEmail(email);
            photo.setAddress(address);
            photo.setPrevious_work_experience(previous_work_experience);
            photo.setPhoto(file.getBytes());
            photo.setContentType(file.getContentType());
            photo.setAuthKey(photo.generateRandomKey());

            photoRepository.save(photo);
        } catch (IOException e) {
            e.printStackTrace();
        }

        return "redirect:/";
    }

    @Transactional(readOnly = true)
    @GetMapping("/photo/{id}")
    @ResponseBody
    public ResponseEntity<byte[]> getPhoto(@PathVariable Long id) {
        Photo photo = photoRepository.findById(id).orElse(null);
        if (photo == null || photo.getPhoto() == null) {
            return ResponseEntity.notFound().build();
        }

        return ResponseEntity.ok()
                .contentType(MediaType.parseMediaType(photo.getContentType()))
                .body(photo.getPhoto());
    }

    @PostMapping("/photo/update/{id}")
    public String updatePhoto(@PathVariable Long id,
                              @RequestParam(value = "photo", required = false)
MultipartFile file,
                              @RequestParam("name") String name,
                              @RequestParam("number_phone") String numberPhone,
                              @RequestParam("email") String email,
                              @RequestParam("address") String address,
                              @RequestParam("previous_work_experience") String
experience) {
        try {
            Photo photo = photoRepository.findById(id).orElse(null);
            if (photo != null) {
                if (file != null && !file.isEmpty()) {
                    photo.setPhoto(file.getBytes());
                    photo.setContentType(file.getContentType());
                }
                photo.setName(name);
                photo.setNumber_phone(numberPhone);
                photo.setEmail(email);
                photo.setAddress(address);
                photo.setPrevious_work_experience(experience);

                photoRepository.save(photo);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return "redirect:/";
```

```
    }

    @PostMapping("/photo/reorder")
    @ResponseBody
    public ResponseEntity<?> reorderPhotos(@RequestBody List<Long> orderedIds) {
        for (int i = 0; i < orderedIds.size(); i++) {
            Photo photo =
photoRepository.findById(orderedIds.get(i)).orElse(null);
            if (photo != null) {
                photo.setPosition(i);
                photoRepository.save(photo);
            }
        }
        return ResponseEntity.ok().build();
    }

    @GetMapping("/adds")
    public String showAddForm() {
        return "adds";
    }

    @GetMapping("/photo/edit/{id}")
    public String editPhotoForm(@PathVariable Long id, Model model) {
        Photo photo = photoRepository.findById(id).orElseThrow(() -> new
IllegalArgumentException("Invalid photo Id:" + id));
        model.addAttribute("photo", photo);
        return "edit-photo";
    }

    @PostMapping("/photo/delete/{id}")
    public String deletePhoto(@PathVariable Long id) {
        workTimeLogRepository.deleteAllByEmployeeId(id);
        photoRepository.deleteById(id);
        return "redirect:/";
    }

    @GetMapping("/photo/export-and-delete/{id}")
    public ResponseEntity<byte[]> exportAndDelete(@PathVariable Long id) {
        byte[] pdfBytes = pdfExportService.generatePdfForEmployee(id);

        workTimeLogRepository.deleteAllByEmployeeId(id);
        photoRepository.deleteById(id);

        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=\"Журнал_відвідувань_" + id + ".pdf\"")
                .contentType(MediaType.APPLICATION_PDF)
                .body(pdfBytes);
    }

    @GetMapping("/photo/export/{id}")
    public ResponseEntity<byte[]> exportWithoutDelete(@PathVariable Long id) {
        byte[] pdfBytes = pdfExportService.generatePdfForEmployee(id);
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=\"Журнал_відвідувань_" + id + ".pdf\"")
                .contentType(MediaType.APPLICATION_PDF)
                .body(pdfBytes);
    }
```

```
    @GetMapping("/")
    public String index(HttpSession session, HttpServletResponse response, Model
model) {
        if (session.getAttribute("adminUnlocked") == null) {
            return "redirect:/admin/unlock";
        }

        // Забороняємо кешування
        response.setHeader("Cache-Control", "no-cache, no-store, must-
revalidate");
        response.setHeader("Pragma", "no-cache");
        response.setDateHeader("Expires", 0);

        List<Photo> photos = photoRepository.findAll(Sort.by("position"));
        model.addAttribute("photos", photos);

        return "index";
    }

}
```

controllers/QrCodeController.java

```
package com.example.final20.controllers;

import com.example.final20.dto.QrCodeService;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.qrcode.QRCodeWriter;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;

import jakarta.annotation.PostConstruct;
import lombok.Getter;
import lombok.Setter;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;


import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;

@Getter
@Setter
@RestController
@RequestMapping("/api")
public class QrCodeController {

    private final QrCodeService qrCodeService;

    public QrCodeController(QrCodeService qrCodeService) {
        this.qrCodeService = qrCodeService;
    }

    @PostConstruct
    public void init() {
```

```
            generateNewQrValue();
        }

        @GetMapping("/qr/current")
        public String getCurrentQrValue() {
            return qrCodeService.getCurrentQrValue();
        }

        @PostMapping("/qr/generate")
        public String generateNewQrValue() {
            return qrCodeService.generateNewQrValue();
        }

        @GetMapping(value = "/qr", produces = MediaType.IMAGE_PNG_VALUE)
        public @ResponseBody byte[] getQrImage() throws Exception {
            String value = qrCodeService.getCurrentQrValue();
            if (value == null) {
                value = qrCodeService.generateNewQrValue();
            }

            int width = 250;
            int height = 250;
            QRCodeWriter qrCodeWriter = new QRCodeWriter();
            BitMatrix bitMatrix = qrCodeWriter.encode(value, BarcodeFormat.QR_CODE,
width, height);

            BufferedImage bufferedImage =
MatrixToImageWriter.toBufferedImage(bitMatrix);
            ByteArrayOutputStream pngOutputStream = new ByteArrayOutputStream();
            ImageIO.write(bufferedImage, "PNG", pngOutputStream);
            return pngOutputStream.toByteArray();
        }
}
```

controllers/SalaryController.java

```
package com.example.final20.controllers;

import com.example.final20.dao.PhotoRepository;
import com.example.final20.dao.SalaryRepository;
import com.example.final20.dao.WorkTimeLogRepository;
import com.example.final20.entities.Photo;
import com.example.final20.entities.Salary;
import com.example.final20.entities.TaxSettings;
import com.example.final20.entities.WorkTimeLog;
import com.example.final20.service.TaxSettingsService;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.List;;
```

```
@Controller
@RequestMapping("/salary")
public class SalaryController {

    private final PhotoRepository photoRepository;
    private final SalaryRepository salaryRepository;
    private final WorkTimeLogRepository workTimeLogRepository;
    private final TaxSettingsService taxSettingsService;
    private final JdbcTemplate jdbcTemplate;

    public SalaryController(PhotoRepository photoRepository,
                            SalaryRepository salaryRepository,
                            WorkTimeLogRepository workTimeLogRepository,
                            TaxSettingsService taxSettingsService, JdbcTemplate
jdbcTemplate) {
        this.photoRepository = photoRepository;
        this.salaryRepository = salaryRepository;
        this.workTimeLogRepository = workTimeLogRepository;
        this.taxSettingsService = taxSettingsService;
        this.jdbcTemplate = jdbcTemplate;
    }

    @PostMapping("/submit")
    public String submitSalary(@RequestParam Long photoId,
                               @RequestParam String salaryType,
                               @RequestParam(required = false) BigDecimal
hourlyRate,
                               @RequestParam(required = false) BigDecimal
dailyRate,
                               @RequestParam BigDecimal bonus,
                               @RequestParam BigDecimal penalty,
                               @RequestParam(required = false) BigDecimal
taxPercent,
                               @RequestParam(required = false) BigDecimal
militaryPercent) {

        if (taxPercent == null || militaryPercent == null) {
            TaxSettings settings = taxSettingsService.getSettings();
            taxPercent = settings.getTaxPercent();
            militaryPercent = settings.getMilitaryTaxPercent();
        }

        Photo employee = photoRepository.findById(photoId).orElseThrow();

        LocalDate firstDayOfMonth = LocalDate.now().withDayOfMonth(1);
        LocalDateTime startOfMonth = firstDayOfMonth.atStartOfDay();
        LocalDateTime now = LocalDateTime.now();

        BigDecimal hoursWorked = BigDecimal.ZERO;
        BigDecimal daysWorked = BigDecimal.ZERO;

        if ("hourly".equalsIgnoreCase(salaryType)) {
            // Обчислити сумарні години
            List<WorkTimeLog> logs =
workTimeLogRepository.findByEmployeeAndCheckInTimeBetween(employee,
startOfMonth, now);

            long totalSeconds = logs.stream()
```

```
                    .filter(log -> log.getCheckInTime() != null &&
log.getCheckOutTime() != null)
                    .mapToLong(log -> Duration.between(log.getCheckInTime(),
log.getCheckOutTime()).getSeconds())
                    .sum();

            hoursWorked = BigDecimal.valueOf(totalSeconds)
                    .divide(BigDecimal.valueOf(3600), 2, RoundingMode.HALF_UP);
        } else if ("daily".equalsIgnoreCase(salaryType)) {
            // Обчислити кількість днів з відмітками (пікання)
            List<WorkTimeLog> logs =
workTimeLogRepository.findByEmployeeAndCheckInTimeBetween(employee,
startOfMonth, now);

            // Визначимо унікальні дати, коли працівник був на роботі
            daysWorked = BigDecimal.valueOf(logs.stream()
                    .filter(log -> log.getCheckInTime() != null)
                    .map(log -> log.getCheckInTime().toLocalDate())
                    .distinct()
                    .count());
        }

        BigDecimal baseSalary;
        if ("hourly".equalsIgnoreCase(salaryType)) {
            baseSalary = hourlyRate.multiply(hoursWorked);
        } else {
            baseSalary = dailyRate.multiply(daysWorked);
        }

        BigDecimal tax =
baseSalary.multiply(taxPercent).divide(BigDecimal.valueOf(100), 2,
RoundingMode.HALF_UP);
        BigDecimal military =
baseSalary.multiply(militaryPercent).divide(BigDecimal.valueOf(100), 2,
RoundingMode.HALF_UP);

        BigDecimal finalSalary =
baseSalary.add(bonus).subtract(penalty).subtract(tax).subtract(military);

        Salary salary = new Salary();
        salary.setEmployee(employee);
        salary.setCalculationDate(LocalDate.now());
        salary.setHoursWorked(hoursWorked);
        salary.setHourlyRate(hourlyRate != null ? hourlyRate : BigDecimal.ZERO);
        salary.setDailyRate(dailyRate != null ? dailyRate : BigDecimal.ZERO);
        salary.setBonus(bonus);
        salary.setPenalty(penalty);
        salary.setTaxPercent(taxPercent);
        salary.setMilitaryTaxPercent(militaryPercent);
        salary.setFinalSalary(finalSalary);
        salary.setDaysWorked(daysWorked.intValue());

        salaryRepository.save(salary);

        return "redirect:/salary/history/" + photoId;
    }

    @GetMapping("/calculate/{id}")
    public String showCalculateForm(@PathVariable Long id, Model model) {
```

```
        Photo employee = photoRepository.findById(id).orElseThrow();
        TaxSettings taxSettings = taxSettingsService.getSettings();

        model.addAttribute("employee", employee);
        model.addAttribute("taxSettings", taxSettings);
        return "salary/calculate";
    }


    @GetMapping("/history/{id}")
    public String viewSalaryHistory(@PathVariable Long id, Model model) {
        Photo employee = photoRepository.findById(id).orElseThrow();
        List<Salary> salaryList = salaryRepository.findByEmployee(employee);
        model.addAttribute("employee", employee);
        model.addAttribute("salaryList", salaryList);
        return "salary/history";
    }

    @PostMapping("/delete/{id}")
    public String deleteSalaryRecord(@PathVariable("id") Long id,
RedirectAttributes redirectAttributes) {
        Salary salary = salaryRepository.findById(id).orElseThrow();
        Long employeeId = salary.getEmployee().getId();
        salaryRepository.deleteById(id);
        redirectAttributes.addFlashAttribute("message", "Запис видалено");
        return "redirect:/salary/history/" + employeeId;
    }
}
```

controllers/ScanController.java

```
package com.example.final20.controllers;

import com.example.final20.dao.PhotoRepository;
import com.example.final20.dto.QrCodeService;
import com.example.final20.dto.ScanRequest;
import com.example.final20.entities.Photo;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.*;

import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/api")
public class ScanController {

    private final JdbcTemplate jdbcTemplate;
    private final PhotoRepository photoRepository;
    private final QrCodeService qrCodeService;

    public ScanController(JdbcTemplate jdbcTemplate, PhotoRepository
```

```
photoRepository, QrCodeService qrCodeService) {
        this.jdbcTemplate = jdbcTemplate;
        this.photoRepository = photoRepository;
        this.qrCodeService = qrCodeService;
    }

    @GetMapping("/user/by-key")
    @ResponseBody
    public ResponseEntity<?> getUserByKey(@RequestParam String key) {
        Photo user = photoRepository.findByAuthKey(key);
        if (user != null) {
            Map<String, Object> response = new HashMap<>();
            response.put("id", user.getId());
            response.put("name", user.getName());
            response.put("email", user.getEmail());
            response.put("number_phone", user.getNumber_phone());

            String photo = "http://192.168.1.20:8080/photo/" + user.getId();
            response.put("photo", photo);

            return ResponseEntity.ok(response);
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Користувача
не знайдено");
        }
    }

    @PostMapping("/scan")
    public ResponseEntity<String> scan(@RequestBody ScanRequest request) {
        String key = request.getEmployee_key();
        String action = request.getAction();
        String qrCode = request.getQrCode();

        List<Integer> ids = jdbcTemplate.queryForList(
                "SELECT id FROM add_photo WHERE auth_key = ?", Integer.class,
key);
        if (ids.isEmpty()) return
ResponseEntity.status(HttpStatus.NOT_FOUND).body("Невірний ключ");

        if (qrCode == null || qrCode.isEmpty()) {
            return ResponseEntity.badRequest().body("QR-код відсутній");
        }

        String cleanQr = qrCode.trim().replaceAll("^'|'$", "");
        if (!cleanQr.equals(qrCodeService.getCurrentQrValue())) {
            return ResponseEntity.badRequest().body("QR-код недійсний або
застарілий");
        }

        int employeeId = ids.get(0);
        LocalDateTime now = LocalDateTime.now();

        if ("check-in".equalsIgnoreCase(action)) {
            jdbcTemplate.update("INSERT INTO work_time_log (employee_id,
check_in_time) VALUES (?, ?)",
                    employeeId, Timestamp.valueOf(now));
            return ResponseEntity.ok("Час приходу збережено");
        } else if ("check-out".equalsIgnoreCase(action)) {
            int updated = jdbcTemplate.update(
```

```
                    "UPDATE work_time_log SET check_out_time = ? WHERE
employee_id = ? AND check_out_time IS NULL ORDER BY check_in_time DESC LIMIT 1",
                    Timestamp.valueOf(now), employeeId);
            if (updated == 0)
                return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Немає
відкритого запису для виходу");
            return ResponseEntity.ok("Час виходу збережено");
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Невірна
дія");
        }
    }
}
```

dao/AdminPinRepository.java

```
package com.example.final20.dao;

import com.example.final20.entities.AdminPin;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AdminPinRepository extends JpaRepository<AdminPin, Long> {
}
```

dao/PhotoRepository.java

```
package com.example.final20.dao;

import com.example.final20.entities.Photo;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PhotoRepository extends JpaRepository<Photo, Long> {
    Photo findByName(String name);
    Photo findByAuthKey(String authKey);
}
```

dao/SalaryRepository.java

```
package com.example.final20.dao;

import com.example.final20.entities.Photo;
import com.example.final20.entities.Salary;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface SalaryRepository extends JpaRepository<Salary, Long> {
    List<Salary> findByEmployee(Photo employee);
}
```

dao/TaxSettingsRepository.java

```
package com.example.final20.dao;

import com.example.final20.entities.TaxSettings;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TaxSettingsRepository extends JpaRepository<TaxSettings, Long>
{
}
```

dao/WorkTimeLogRepository.java

```
package com.example.final20.dao;

import com.example.final20.entities.Photo;
import com.example.final20.entities.WorkTimeLog;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.time.LocalDateTime;
import java.util.List;

@Repository
public interface WorkTimeLogRepository extends JpaRepository<WorkTimeLog, Long>
{
    List<WorkTimeLog> findByEmployee(Photo employee);

    @Query("SELECT w FROM WorkTimeLog w WHERE w.employee.id = :empId")
    List<WorkTimeLog> findAllByEmployeeId(@Param("empId") Long empId);

    @Transactional
    @Modifying
    @Query("DELETE FROM WorkTimeLog w WHERE w.employee.id = :employeeId")
    void deleteAllByEmployeeId(@Param("employeeId") Long employeeId);
    List<WorkTimeLog> findByEmployeeAndCheckInTimeBetween(Photo employee,
LocalDateTime start, LocalDateTime end);
}
```

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

1 Загальні положення

Ця інструкція призначена для користувачів програмного забезпечення, розробленого з метою автоматизації нарахування заробітної плати в кав'ярні Lite Cafe. Програмне забезпечення складається з двох частин: вебінтерфейсу для адміністратора та мобільної частини для працівників.

2 Вимоги до користувача

Для адміністратора: базові навички роботи з комп'ютером та браузером.

Для працівника: наявність мобільної частини WorkTimeApp з операційною системою Android.

3 Вхід до системи

3.1 Вхід до системи для адміністратора

Запустіть вебінтерфейс у браузері (наприклад: Google Chrome).

Увійдіть, ввівши електронну адресу резервний PIN-код та новий PIN-код (рисунок В.1). Після чого, користувачеві надійде лист з підтвердженням електронної пошти (рисунок В.2).

У разі втрати PIN-коду скористайтесь функцією «Забули PIN-код». Після чого з’явиться варіанти зміни PIN-коду (рисунок В.3). PIN-код можна відновити за допомогою резервного PIN-коду (рисунок В.4), але якщо забули резервний PIN-код, то можна відновити за допомогою функції «Надіслати електронного листа» На відновлення PIN-коду надійде лист на електрону адресу, яка була введена на початку авторизації. Також можна змінити PIN-код, якщо попередній був не надійний (рисунок В.5).

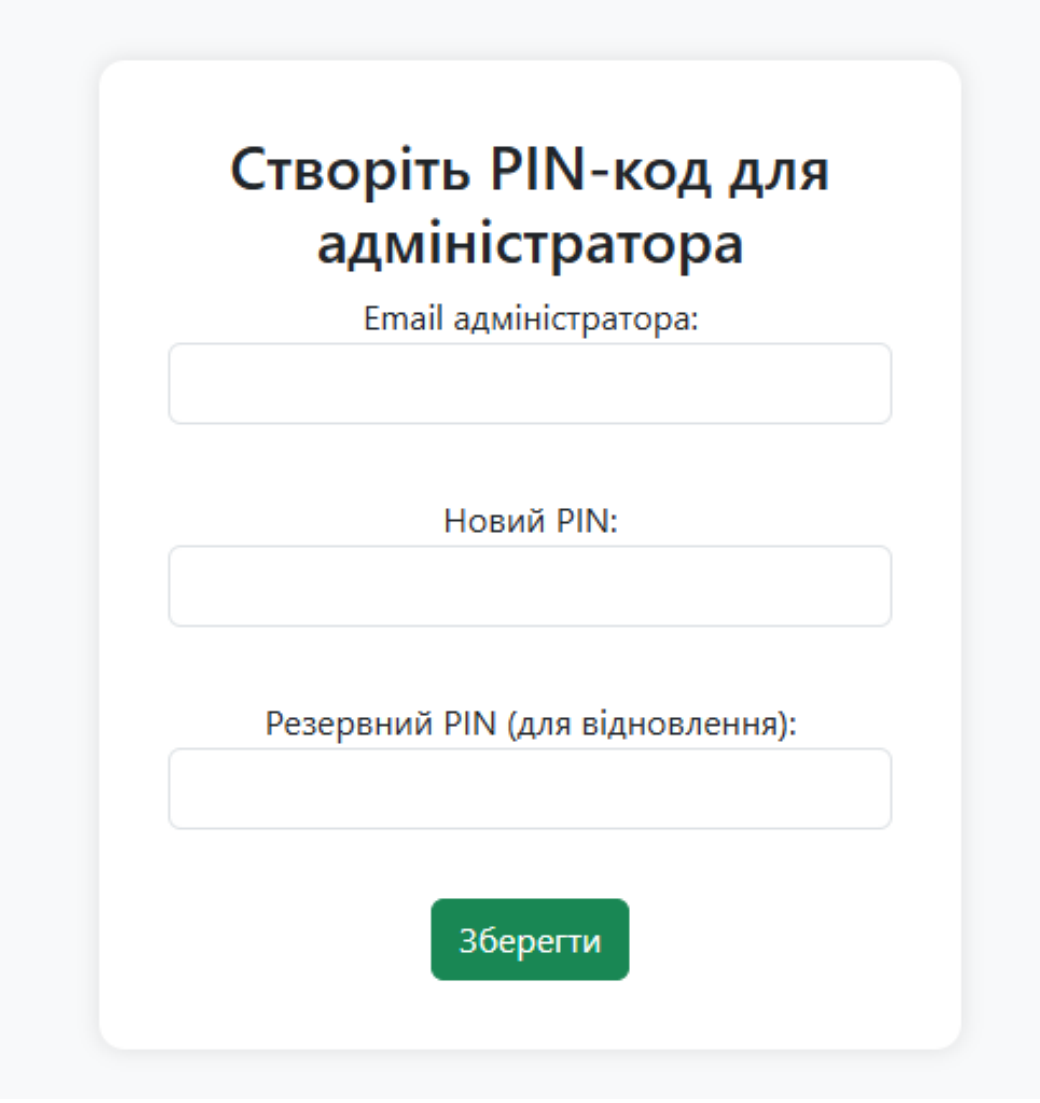

Рисунок В.1 – Початкова сторінка

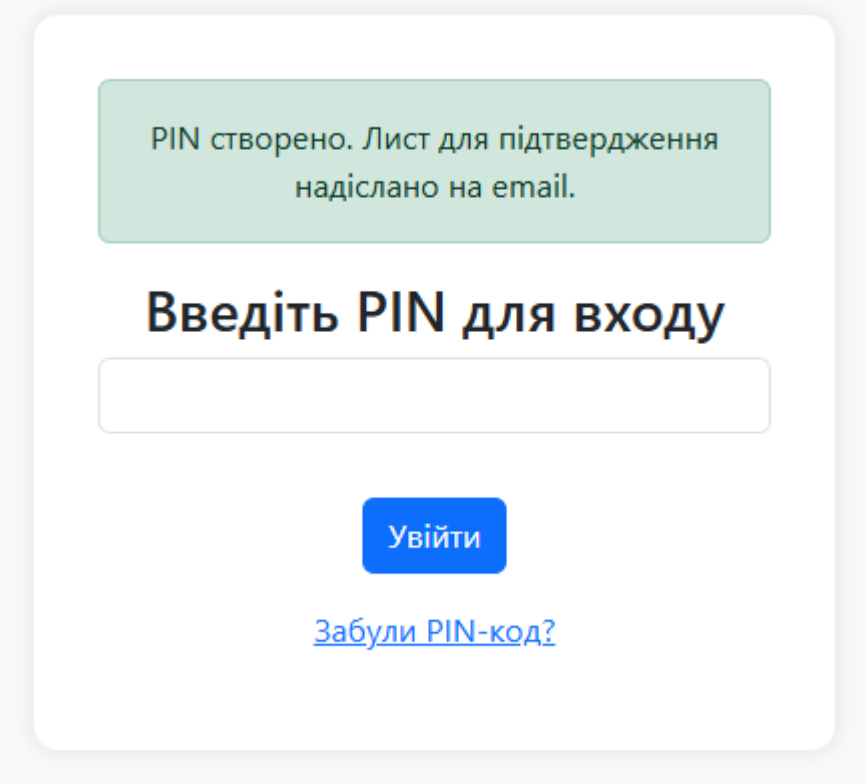

Рисунок В.2 – Повідомлення про надіслання листа для підтвердження

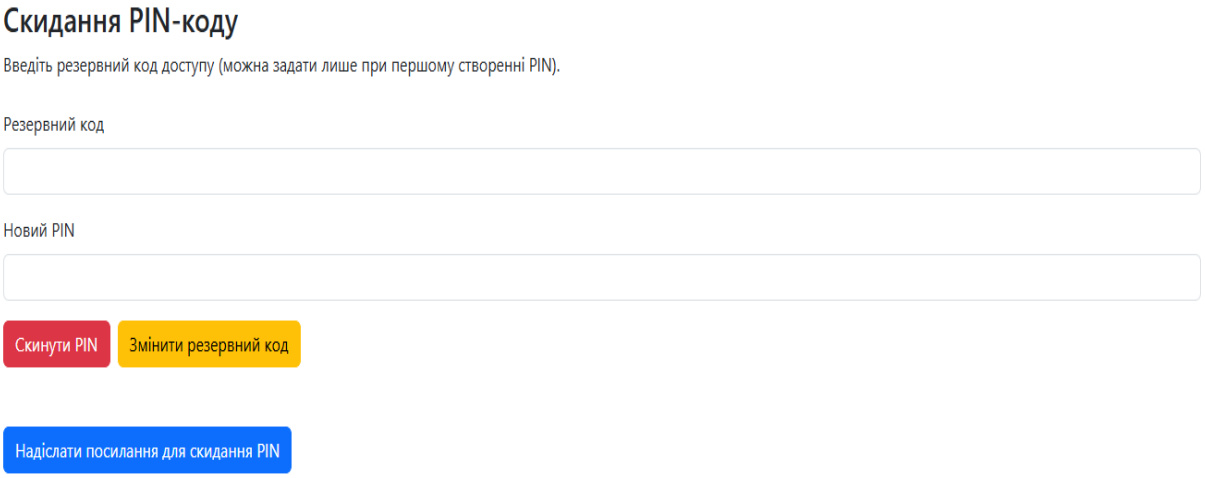

Рисунок В.3 – Сторінка скидання PIN-коду

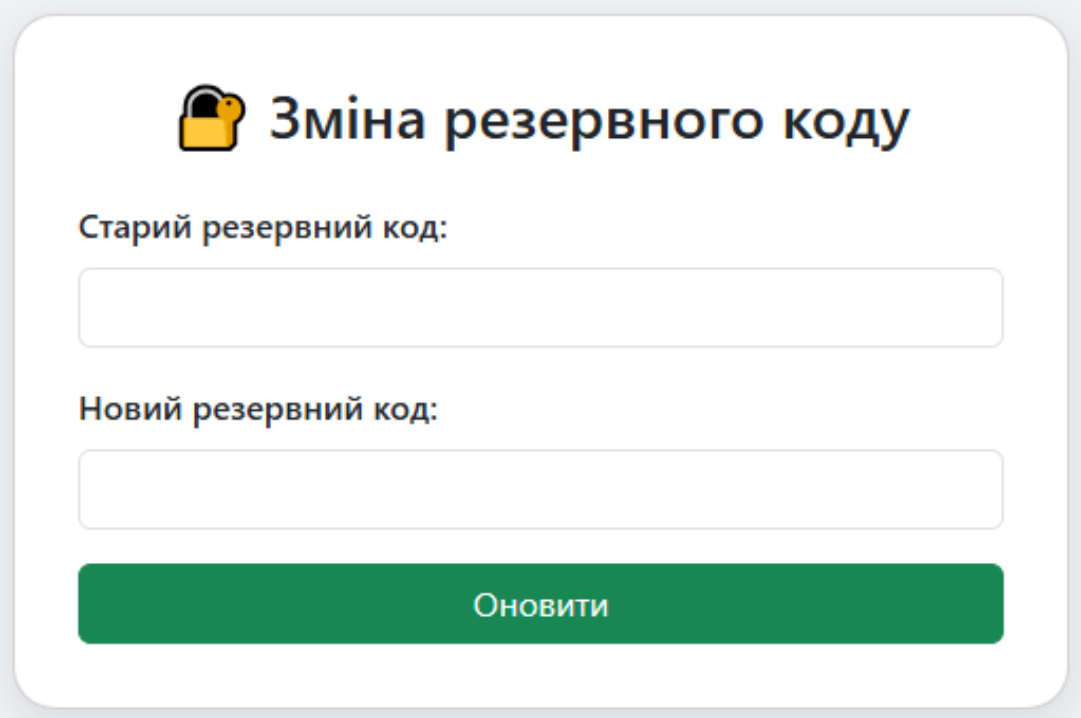

Рисунок В.4 – Сторінка зміна резервного коду

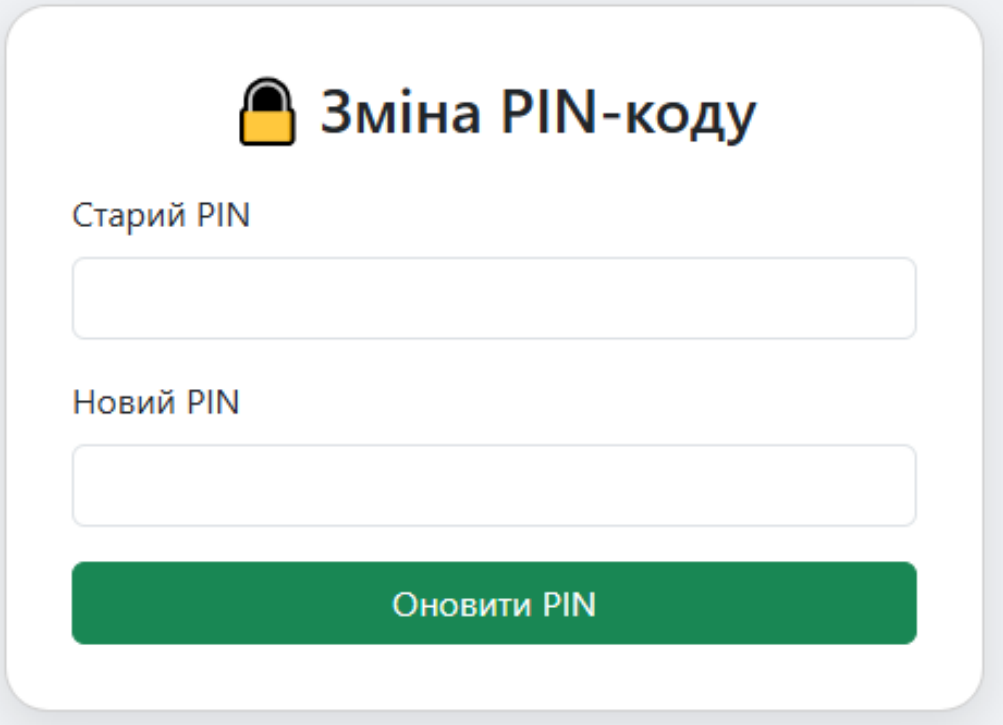

Рисунок В.5 – Сторінка зміна PIN-коду

3.2 Вхід до системи для працівника

Відкрийте телефонну частину програмного забезпечення (рисунок В.6).

Скануйте QR-код, що знаходиться на робочому місці або введіть ідентифікаційний ключ (рисунок В.7).

Після сканування, працівника перекине на головну сторінку мобільної частини (рисунок В.8), де зможе сканувати універсальний QR-код для приходу та виходу на роботі (рисунок В.9). Програмне забезпечення автоматично відмітить прихід або вихід з роботи (рисунок В.10). Працівник сканується за допомогою камери де може вийти з камери або увімкнути ліхтарик (рисунок В.11).

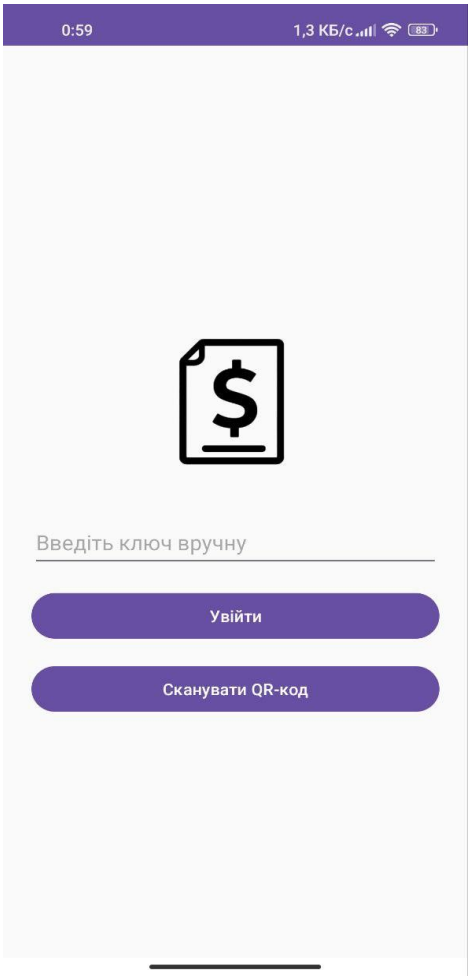

Рисунок В.6 – Сторінка мобільної частини програмного забезпечення

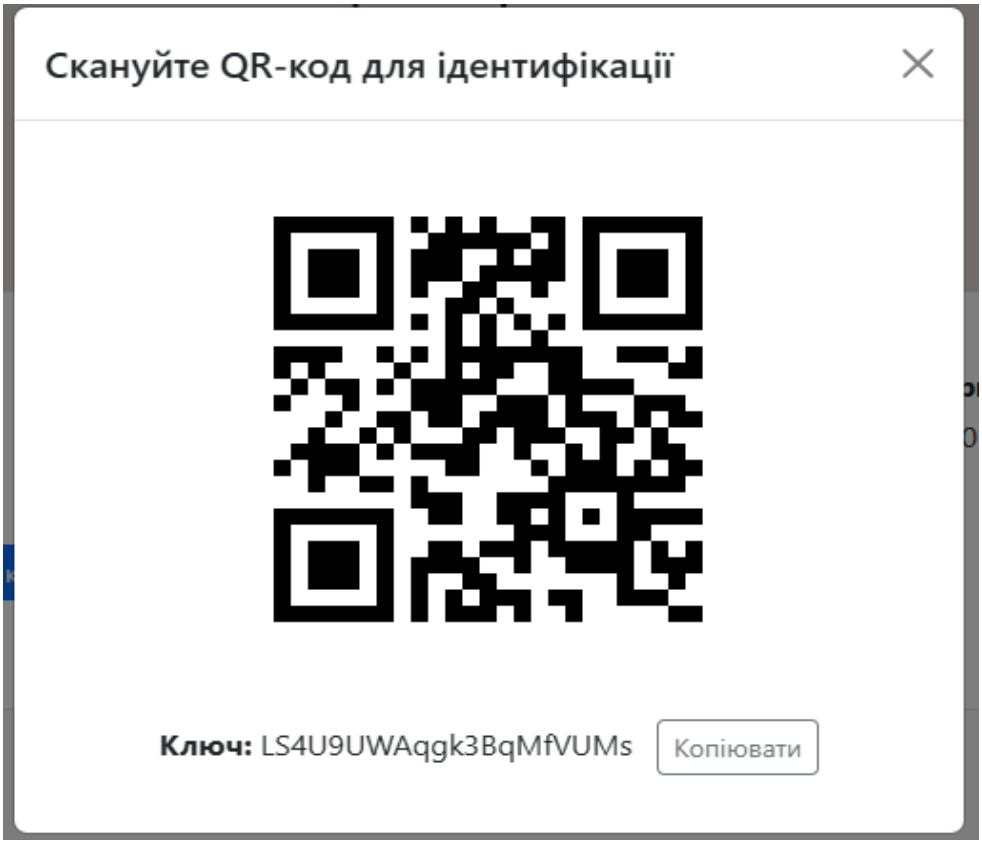

Рисунок В.7 – Модульне вікно з для ідентифікації та прихованим ключом для ручного ідентифікування працівника в мобільній частині

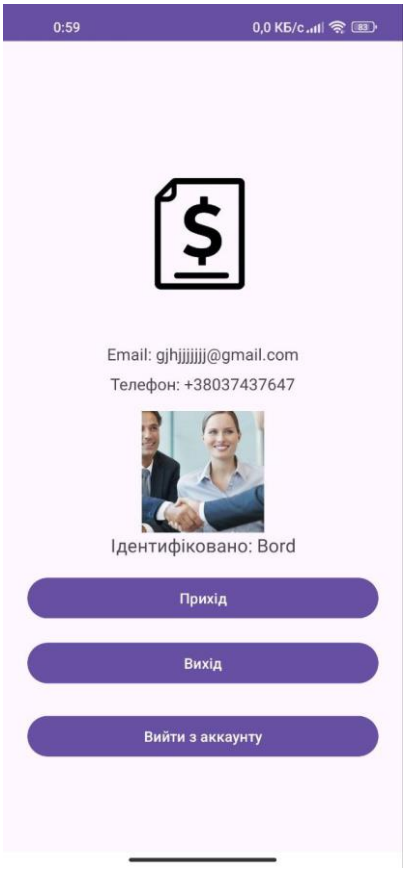

Рисунок В.8 – Головна сторінка мобільної частини програмного забезпечення

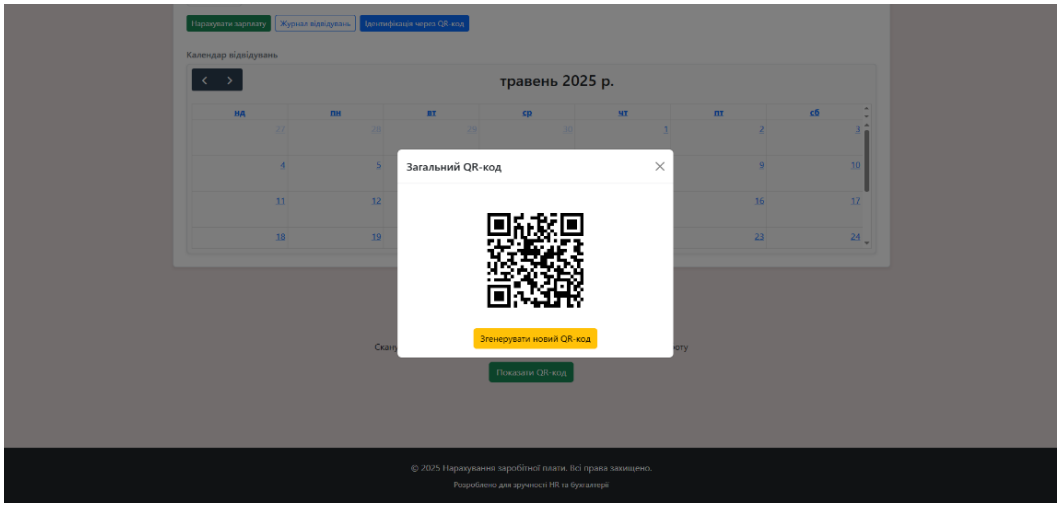

Рисунок В.9 – Універсальний QR-код для сканування приходу та виходу на роботу, а також генерування нового QR-коду на випадок, якщо хтось фотографує

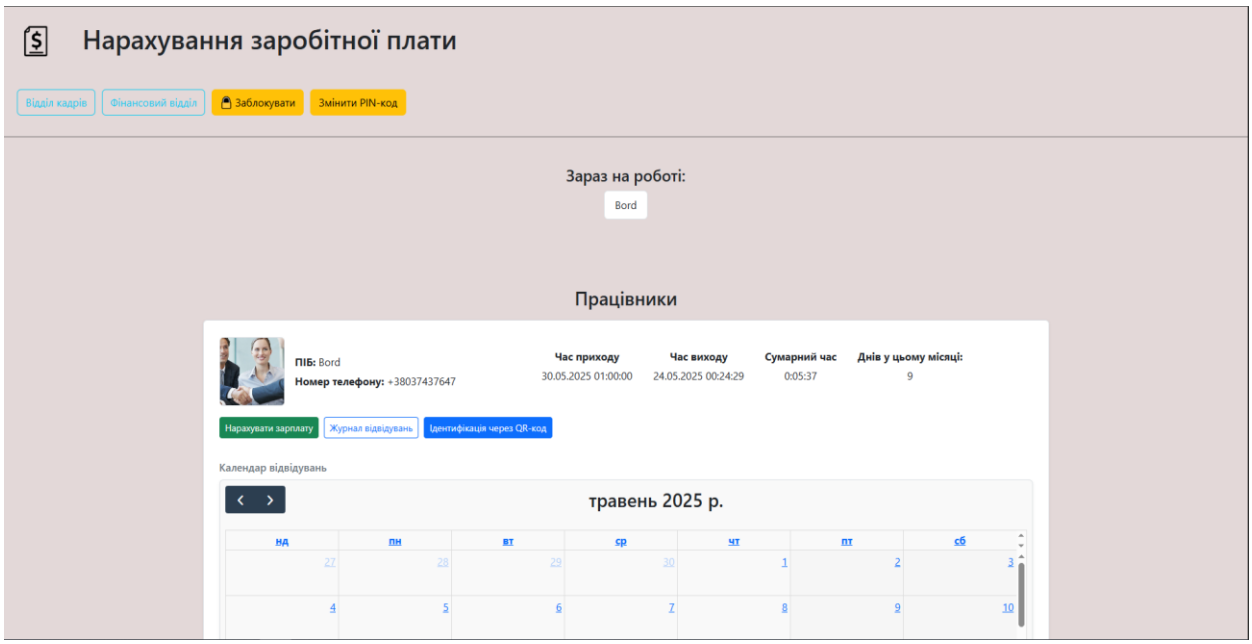

Рисунок В.10 – Працівник сканувався на роботу

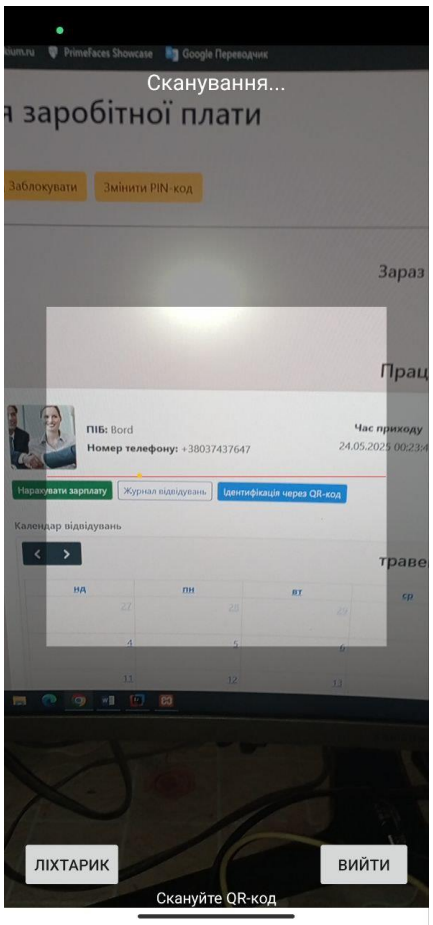

Рисунок В.11 – Камера для сканування

4 Управління працівниками

У вебінтерфейсу адміністратора:

- Для додавання працівника, натисніть кнопку «Додати працівника», заповніть необхідні поля (ПІБ, телефон, електрону адресу, фото, досвід тощо) та збережіть (рисунок В.12).
- Для редагування працівника, натисніть на запис працівника, вносіть зміни та збережіть (рисунок В. 13).
- Для видалення працівника, натисніть кнопку «Видалити» у вікні редагування, а також можливість зберегти історію відвідування (рисунок В.14).

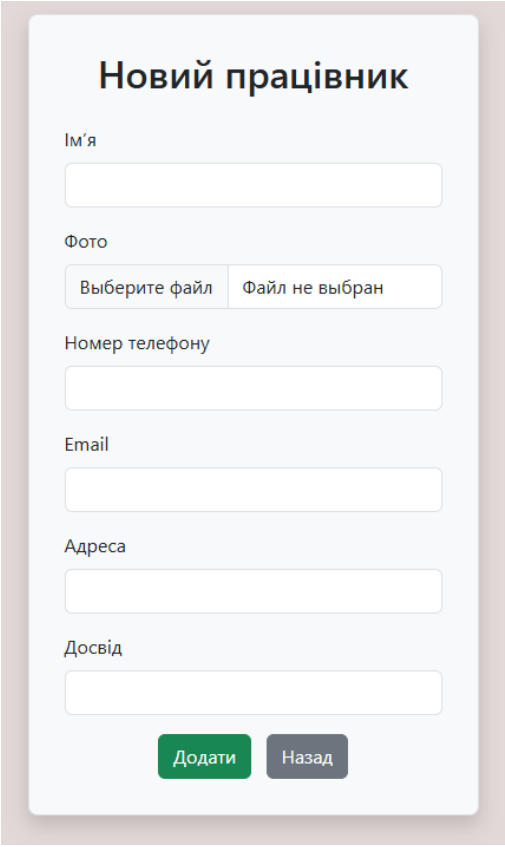

Рисунок В.12 – Сторінка додавання працівника

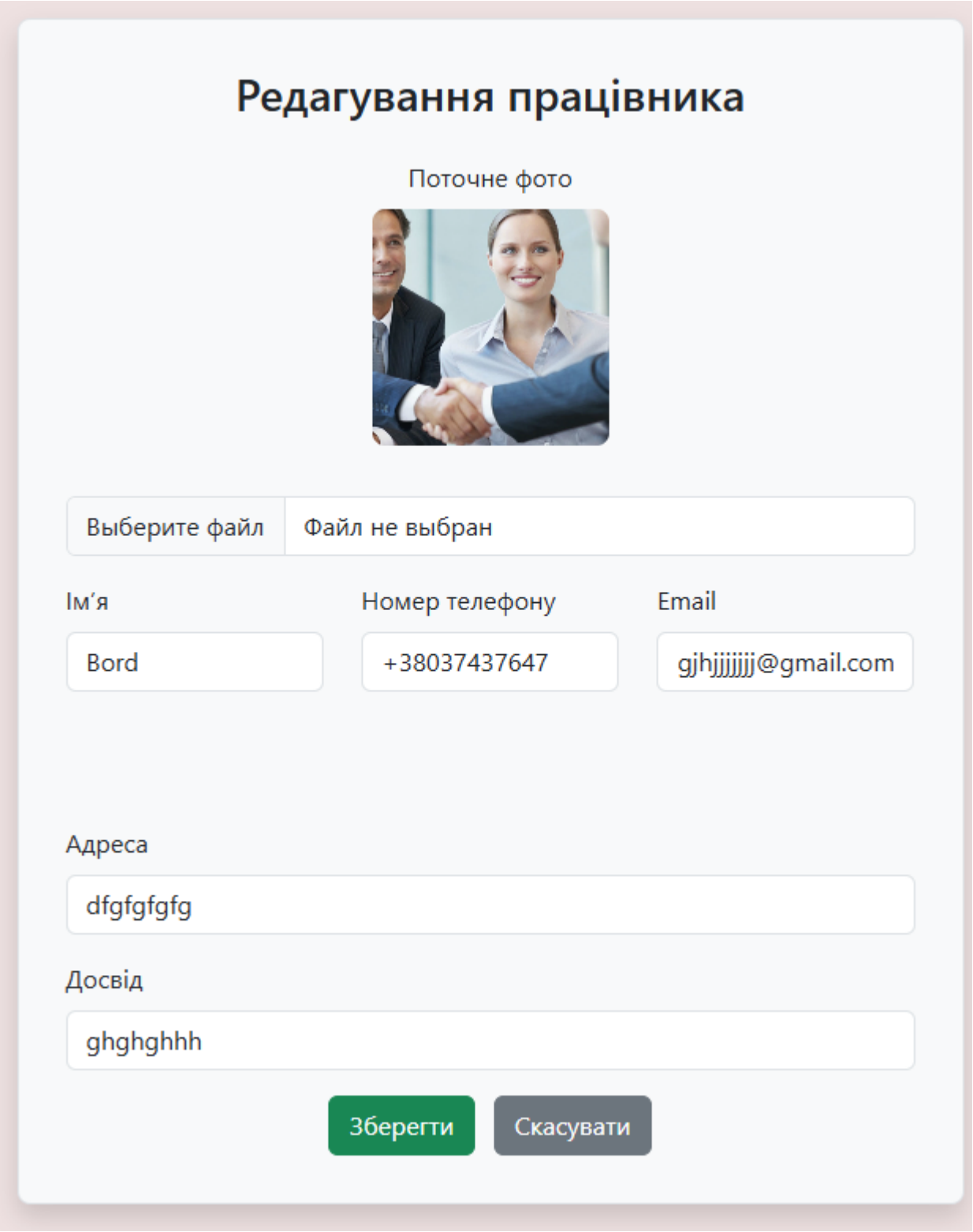

Рисунок В.13 – Сторінка редагування працівника

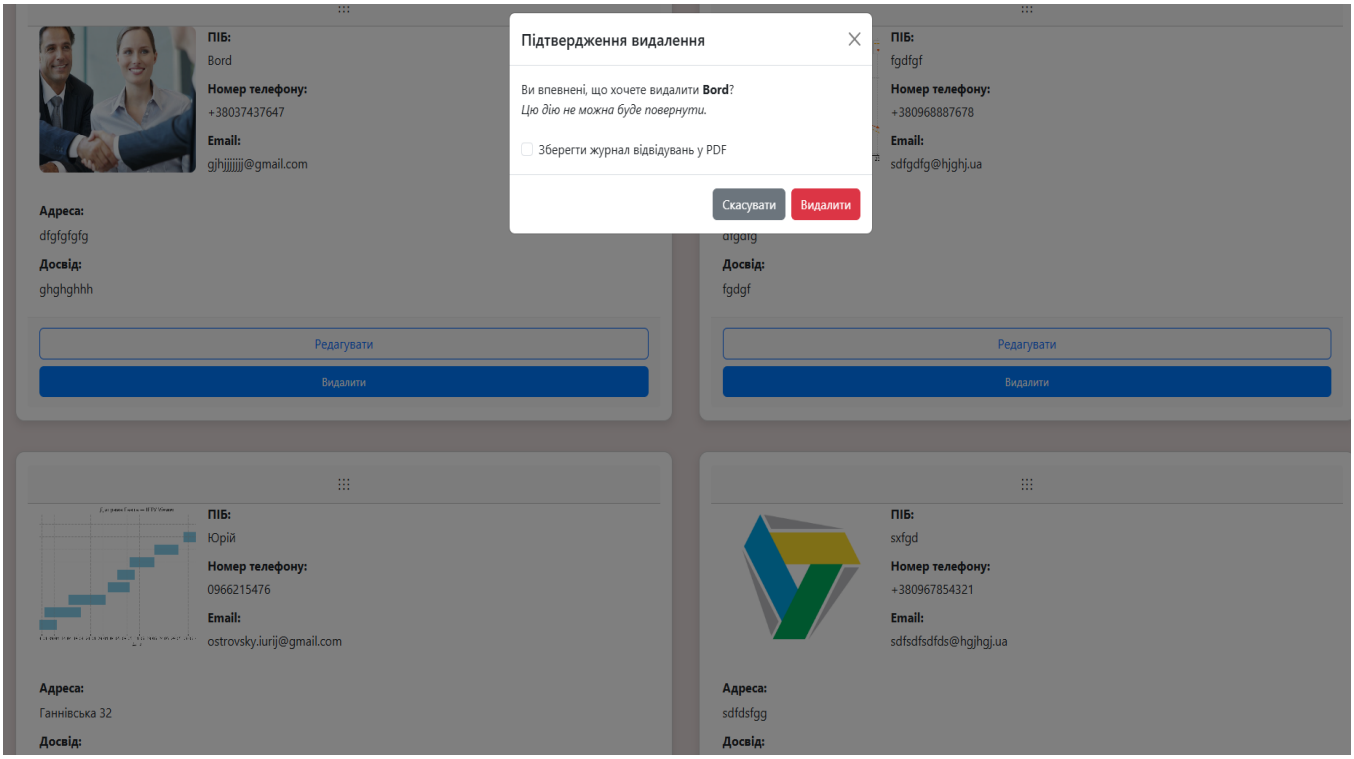

Рисунок В.14 – Модульне вікно для видалення працівника, а також функція зберігання відвідувань

5 Фіксація робочого часу

Усі сканування QR-коду працівниками автоматично фіксуються в системі. Для перегляду історії відвідувань відкрийте розділ «Журнал відвідувань» або «Календар змін» (рисунки В.15 – В.16).

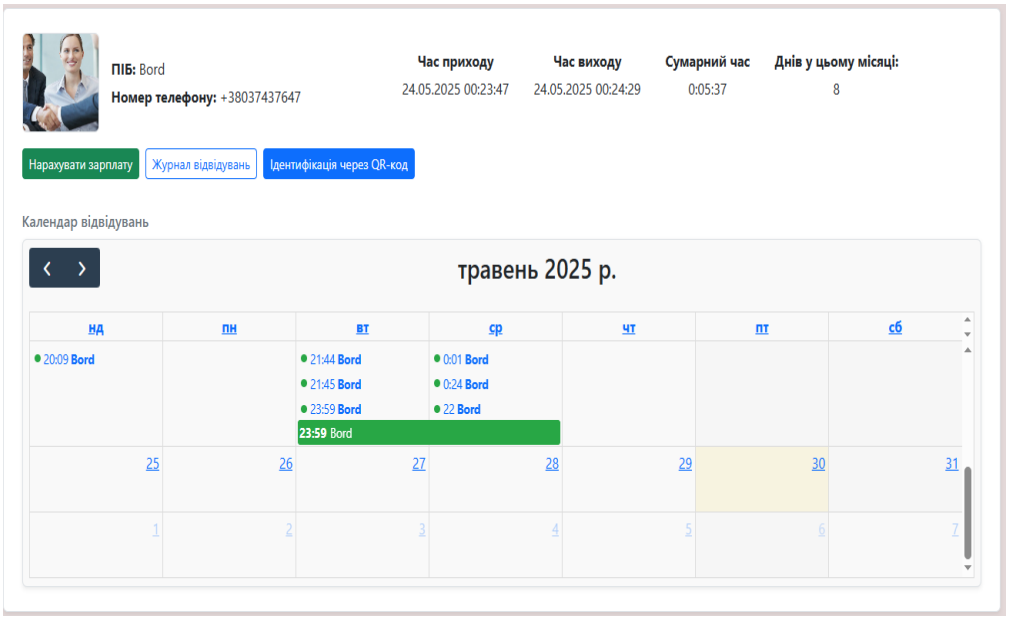

Рисунок В.15 – Календар відвідування працівника

Рисунок В.16 – Журнал відвідування працівника

6 Розрахунок заробітної плати

1) Перейдіть у розділ «Фінансовий відділ».

2) Оберіть працівника.

3) Вкажіть тип ставки (погодинна або денна), премії, штрафи та податкові коефіцієнти (рисунки В.17 – В.18).

4) Натисніть кнопку «Нарахувати».

5) Результат розрахунку буде збережено на екрані або у звіті, який можна переглянути чи роздрукувати (рисунки В.19 – В.21).

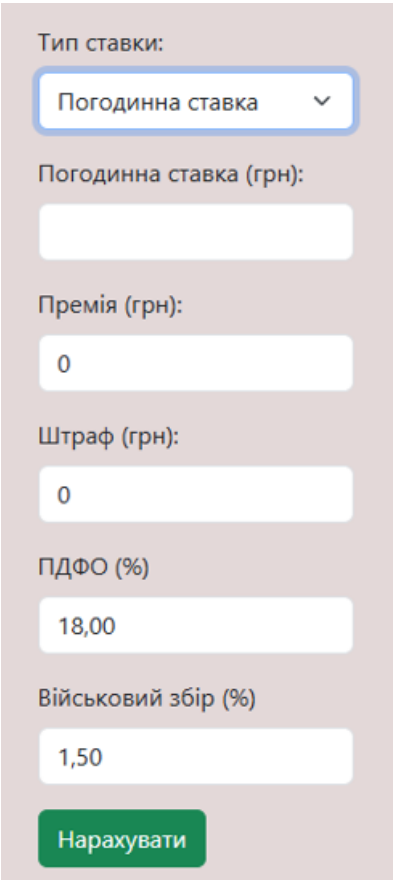

Рисунок В.17 – Сторінка нарахування заробітної плати

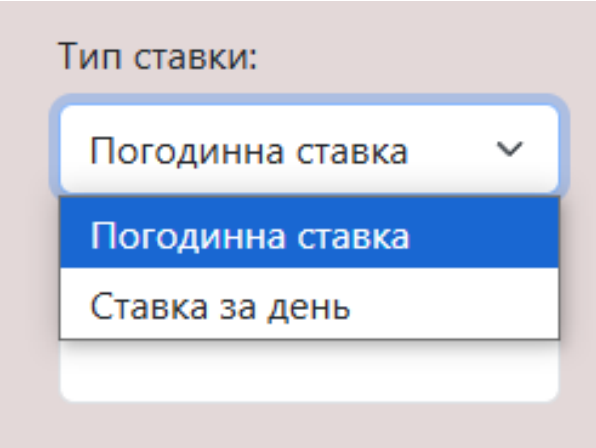

Рисунок В.18 – Типи ставок

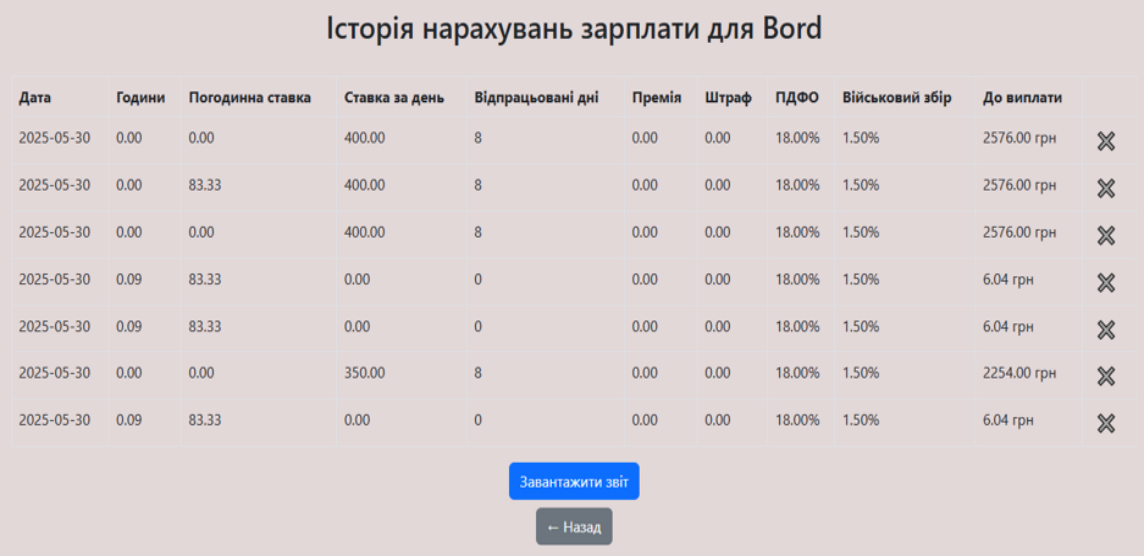

Рисунок В.19 – Сторінка розрахунку заробітної плати працівника

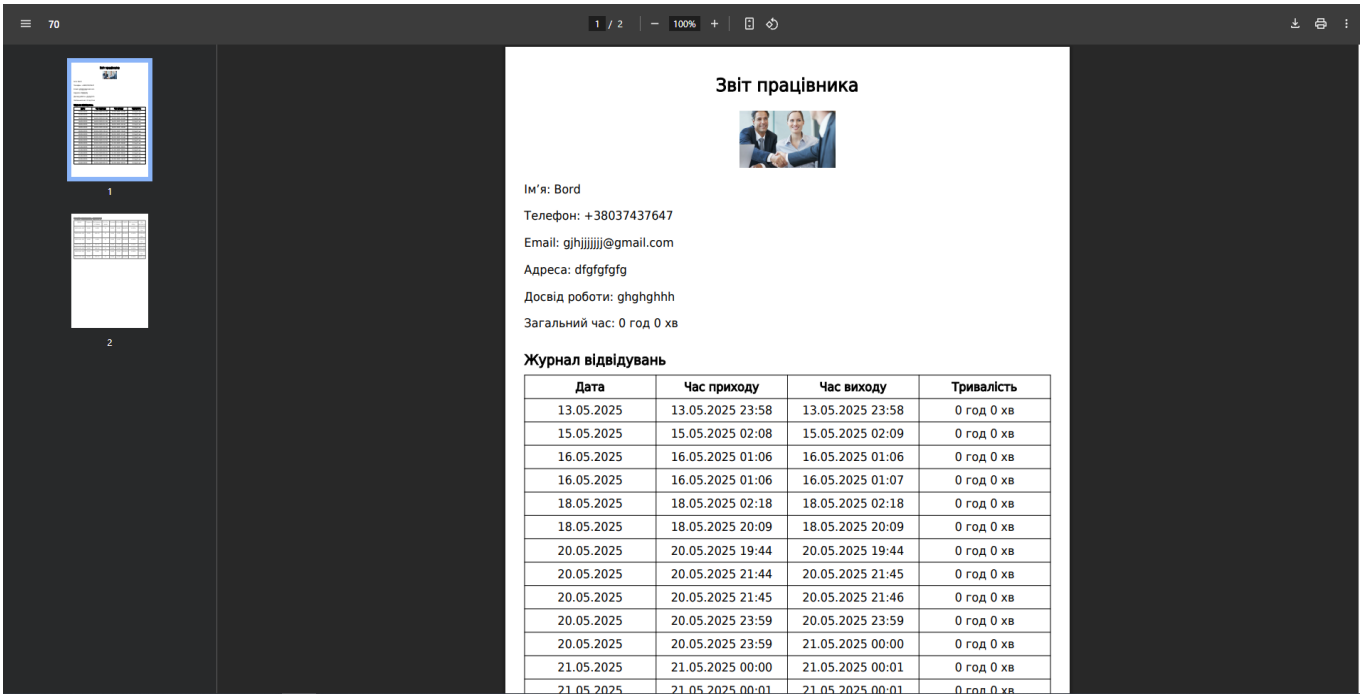

Рисунок В.20 – Звіт про розрахунок (перша частина)

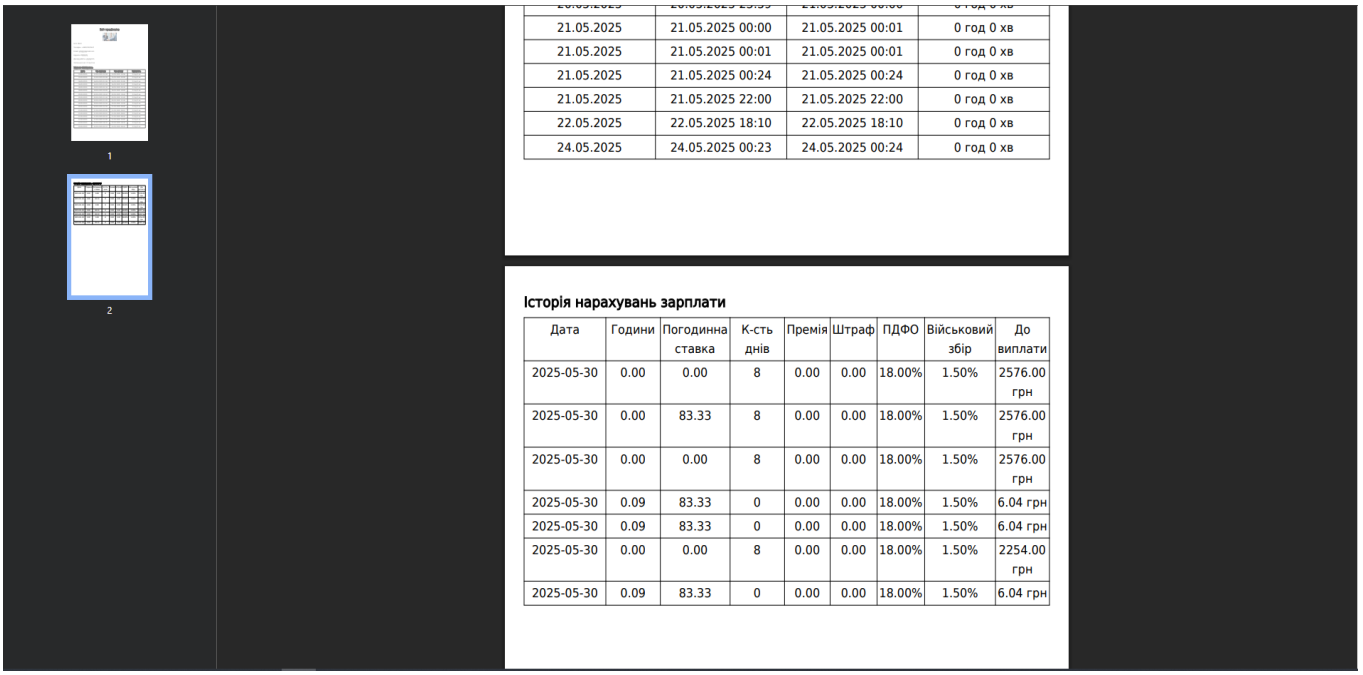

Рисунок В.21 – Звіт про розрахунок (друга частина)

ДОДАТОК Г – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

Вступ

Найменування програми: Програмне забезпечення для автоматизації нарахування заробітної плати кав'ярні Lite Cafe.

Позначення програми: Програмне забезпечення «LC-SalarySystem».

1 Функціональне призначення

1.1 Класи розв'язуваних завдань та призначення програми

Розроблене програмне забезпечення призначене для автоматизації нарахування заробітної плати працівникам кав'ярні Lite Cafe. Воно дозволяє зменшити ручну працю адміністратора, знизити ризик помилок при розрахунках, забезпечити надійність виплат.

Програмне забезпечення виконує такі основні завдання:

- збирання та збереження даних працівників;
- облік відпрацьованого часу;
- нарахування заробітної плати з урахуванням податків, штрафів та премій;
- генерація звітності для адміністрації.

1.2 Функціональні обмеження на застосування

Програма призначена для використання в межах одного закладу малого бізнесу. Програмне забезпечення не підтримує обробку великого обсягу даних (наприклад, понад 100 співробітників), а також не передбачає інтеграцію з державними системами звітності чи банківськими платформами.

2 Опис логічної структури

2.1 Опис структури

Структура програмного забезпечення побудована за принципом Model-View-Controller (MVC), що забезпечує чітке розділення логіки, інтерфейсу та доступу до даних.

- Model — відповідає за роботу з базою даних, зокрема облік працівників, відпрацьованих годин, розрахунок зарплати.
- View — реалізує користувацький інтерфейс у вигляді веб сторінки та мобільного додатку.
- Controller — керує логікою взаємодії між Model та View.

2.2 Мова програмування

Для реалізації серверної частини використано JavaScript, Java, клієнтська частина — HTML, CSS. Телефонна частина програмного забезпечення реалізовано на Kotlin (Android Studio).

2.3 Вхідні та вихідні дані

Вхідні дані: інформація про працівників, час приходу та виходу, ставка, коефіцієнти премій і штрафів.

Вихідні дані: розрахована заробітна плата, таблиці обліку часу, звіти про виплати.

3 Склад і функції

3.1 Склад програмного забезпечення

- Вебінтерфейс для адміністратора;
- База даних MariaDB;
- Телефонна частина програмного забезпечення для працівників;
- Серверна частина (API).

3.2 Функції програмного забезпечення

- Додавання/редагування даних працівників;
- Фіксація відпрацьованого часу за допомогою QR-коду;
- Розрахунок зарплати;
- Захист даних адміністратора (PIN-код);
- Формування звітів.

4 Умови застосування

4.1 Вимоги до складу та параметрів технічних засобів

Сервер має наступні мінімальні технічні та системні параметри:

- ОС: Linux або Windows;
- CPU: не нижче 2.0 GHz;
- RAM: від 2 ГБ;
- ПЗ: Node.js, PostgreSQL.

Клієнтська частина повинна мати наступні мінімальні технічні та системні параметри:

- Мобільний пристрій з Android 8+;

4.2 Вимоги до програмної сумісності

Програмне забезпечення розроблено таким чином, щоб працювати на популярних сучасних платформах. Серверна частина сумісна з операційними системами Linux (наприклад, Ubuntu) та Windows. Для зберігання даних використовується база даних MariaDB.

Вебінтерфейс програми відкривається у сучасних браузерах, таких як Google Chrome, Mozilla Firefox або Microsoft Edge. Для коректної роботи рекомендується використовувати актуальні версії цих браузерів.

Телефонна частина програмного забезпечення працює на Android-пристроях з версією системи не нижче Android 8.0. Обмін даними між сервером і клієнтом відбувається через REST API з використанням формату JSON.

Уся програмне забезпечення підтримує українську мову та UTF-8-кодування, що забезпечує правильне відображення тексту.

4.3 Спосіб виклику програми

Виклик програмного забезпечення відбувається через браузер. Вхід адміністратора захищено PIN-кодом.

Вхідною точкою в програмне забезпечення є головна сторінка. Для працівників телефонна частина програмного забезпечення.

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАРАХУВАННЯ ЗАРОБІТНОЇ ПЛАТИ КАВ'ЯРНІ LITE CAFE

1 Об'єкт випробувань

Об'єктом випробувань є програмне забезпечення для автоматизації нарахування заробітної плати у кав'ярні Lite Cafe. Програмне забезпечення включає серверну частину, вебінтерфейс для адміністратора та телефонна частина програмного забезпечення для працівників.

2 Мета випробувань

Метою проведення випробувань є перевірка відповідності програмного забезпечення технічному завданню, правильності реалізації всіх функцій, стабільності роботи та зручності використання.

3 Вимоги до програми

При проведенні випробувань необхідно перевірити, що розроблене програмне забезпечення реалізує наступні функції:

- Правильність фіксації часу приходу та виходу працівника;
- точність розрахунку заробітної плати (з урахуванням ставки, податків, премій, штрафів);
- збереження та відображення персональних даних;

- безперебійна робота мобільного додатку;
- генерація звітів без помилок;
- захист доступу до адміністративної частини програми.

4 Вимоги до програмної документації

До складу програмної документації, що надається на випробування входить:

- технічне завдання;
- текст програми;
- інструкція користувача;
- опис програми;
- програма і методика випробувань.

5 Засоби та порядок випробувань

5.1 Засоби випробувань

До складу технічних засобів, які мають використовуватися при випробуваннях належать:

- Комп’ютер з ОС Windows 10.

До складу програмних засобів, які мають використовуватися при випробуваннях належать:

- Телефонна частина програмного забезпечення;
- браузер Google Chrome.

5.2 Порядок проведення випробувань

1) Перевірка комплектності програмної документації;
2) Перевірка функції фіксації робочого часу через телефонну частину програмного забезпечення;
3) Перевірка функції розрахунку заробітної плати;
4) Перевірка функції авторизації;

6 Методи випробувань

1) Випробування функції розрахунку заробітної плати
   Метод: ручне введення тестових даних (ставка, кількість годин, премія, штраф) та порівняння результату з очікуваним.
   Очікуваний результат: правильна сума виплати.
2) Випробування функції QR-сканування на мобільному додатку
   Метод: сканування тестового QR-коду.
   Очікуваний результат: поява працівника в списку «Зараз на роботі».
3) Випробування функції авторизації адміністратора
   Метод: вхід до панелі з правильним та неправильним PIN-кодом.
   Очікуваний результат: доступ надається тільки при вірному PIN-коді.